

## **EXHIBIT A**

## CLAIMS AS ORIGINALLY FILED

1. In a computerized system that includes a content server, a mobile gateway, and a first and a second mobile client, the first and second mobile clients differing from each other in at least one operating characteristic, wherein the mobile gateway receives content that is addressed to the first and second mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client and assigning a second transform to the second mobile client, the first and second transforms specifically considering one or more operating characteristics of the first and second mobile clients;

receiving content from the content server;

altering the content according to the first and second transforms so that the content is compatible with the one or more operating characteristics of the first and second mobile clients, the altered content comprising a first transformed content and a second transformed content;

establishing a communication link between the mobile gateway and the first and second mobile clients; and

sending the first transformed content to the first mobile client and sending the second transformed content to the second mobile client.

2. A method as recited in claim 0 further comprising the act of at least one of the transforms encrypting the content.

3. A method as recited in claim 0 further comprising the act of at least one of the transforms compressing the content.

4. A method as recited in claim 0 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, and a cascaded mobile gateway.

5. A method as recited in claim 0 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

6. A method as recited in claim 0 wherein the one or more operating characteristics considered by the first and second transforms include at least one of the first and second mobile client's software, processor, memory, display, and communication link.

7. A method as recited in claim 0 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

8. A method as recited in claim 0 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

9. A method as recited in claim 0 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, and sports content.



10. In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content to the first mobile client.

11. A method as recited in claim 10 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, and communication link.

12. A method as recited in claim 10 further comprising the act of the first transform encrypting the content.

13. A method as recited in claim 10 further comprising the act of the first transform compressing the content.

14. A method as recited in claim 10 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

15. A method as recited in claim 10 wherein the first mobile client is one of a telephone, a pager, a personal digital assistant, and a cascaded mobile gateway.

16. A method as recited in claim 10 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, and sports content.

17. A method as recited in claim 10 wherein the computerized system includes a second mobile client, the method further comprising the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

18. A method as recited in claim 17 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

19. A method as recited in claim 17 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

20. In a computerized system that includes a content server, a mobile gateway, and mobile clients, wherein some of the mobile clients differ from each other in at least one operating characteristic, and wherein the mobile gateway receives content that is addressed to the mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising the mobile gateway performing steps for:

associating content transforms with a first and a second mobile client, the content transforms accounting for one or more operating characteristics of the first and second mobile clients;

producing first transformed content and second transformed content based on content from the content server and the content transforms; and

providing the first and second transformed content to the first and second mobile clients.

21. A method as recited in claim 20 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, and a cascaded mobile gateway.

22. A method as recited in claim 20 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising steps for:

producing a first additional transformed content based on a content transform associated with the first mobile client; and

providing the first additional transformed content to the first mobile client.

23. A method as recited in claim 20 wherein the one or more operating characteristics considered by the content transforms include at least one of the mobile clients' software, processor, memory, display, and communication link.

24. A method as recited in claim 20 wherein the computerized system includes a third mobile client, the method further comprising a step for providing the first transformed content to the third mobile client, due to similarities in one or more operating characteristics of the first and third mobile clients.

25. A method as recited in claim 20 wherein the content received from the content server is addressed to a list of mobile clients, the method further comprising a step for providing the content to each of the specific mobile clients contained in the list.

26. A method as recited in claim 20 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, and sports data.

27. In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a computer program product for implementing a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method at a mobile gateway; and

wherein said method is comprised of machine-executable instructions for performing the acts of:

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content to the first mobile client.

28. A computer program product as recited in claim 27 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, and communication link.

29. A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of encrypting the content.

30. A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of compressing the content.

31. A method as recited in claim 27 wherein the first transformed content comprises a notification that additional content is available at the content server, the method comprised further of machine-executable instructions for performing the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

32. A computer program product as recited in claim 27 wherein the first mobile client is one of a telephone, a pager, and a personal digital assistant and wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, and sports data.

33. A computer program product as recited in claim 27 wherein the computerized system includes a second mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

34. A computer program product as recited in claim 33 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method comprised further of machine-executable instructions for performing the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.



35. A computer program product as recited in claim 33 wherein the computerized system includes a third mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

## CLAIMS AFTER AMENDMENT A

1. (Original) In a computerized system that includes a content server, a mobile gateway, and a first and a second mobile client, the first and second mobile clients differing from each other in at least one operating characteristic, wherein the mobile gateway receives content that is addressed to the first and second mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client and assigning a second transform to the second mobile client, the first and second transforms specifically considering one or more operating characteristics of the first and second mobile clients;

receiving content from the content server;

altering the content according to the first and second transforms so that the content is compatible with the one or more operating characteristics of the first and second mobile clients, the altered content comprising a first transformed content and a second transformed content;

establishing a communication link between the mobile gateway and the first and second mobile clients; and

sending the first transformed content to the first mobile client and sending the second transformed content to the second mobile client.

2. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms encrypting the content.

3. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms compressing the content.

4. (Currently Amended) A method as recited in claim 1 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, ~~and~~ or a cascaded mobile gateway.

5. (Original) A method as recited in claim 1 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

- receiving a request for the additional content from the first mobile client,
- retrieving the additional content from the content server;
- altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and
- sending the first transformed additional content to the first mobile client.

6. (Currently Amended) A method as recited in claim 1 wherein the one or more operating characteristics considered by the first and second transforms include at least one of the first ~~and~~or second mobile client's software, processor, memory, display, ~~and~~or communication link.

7. (Original) A method as recited in claim 1 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

- assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and
- sending the first transformed content to the third mobile client.

8. (Original) A method as recited in claim 1 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

9. (Currently Amended) A method as recited in claim 1 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, ~~and~~or sports content.

10. (Original) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content to the first mobile client.

11. (Currently Amended) A method as recited in claim 10 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, ~~and~~ or communication link.

12. (Original) A method as recited in claim 10 further comprising the act of the first transform encrypting the content.

13. (Original) A method as recited in claim 10 further comprising the act of the first transform compressing the content.

14. (Original) A method as recited in claim 10 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

- receiving a request for the additional content from the first mobile client,
- retrieving the additional content from the content server;
- altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and
- sending the first transformed additional content to the first mobile client.

15. (Currently Amended) A method as recited in claim 10 wherein the first mobile client is one of a telephone, a pager, a personal digital assistant, ~~and~~or a cascaded mobile gateway.

16. (Currently Amended) A method as recited in claim 10 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, ~~and~~or sports content.

17. (Original) A method as recited in claim 10 wherein the computerized system includes a second mobile client, the method further comprising the acts of:

- assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;
- altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;
- establishing a communication link between the mobile gateway and the second mobile client; and
- sending the second transformed content to the second mobile client.

18. (Original) A method as recited in claim 17 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

19. (Original) A method as recited in claim 17 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client;  
and

sending the first transformed content to the third mobile client.

20. (Original) In a computerized system that includes a content server, a mobile gateway, and mobile clients, wherein some of the mobile clients differ from each other in at least one operating characteristic, and wherein the mobile gateway receives content that is addressed to the mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising the mobile gateway performing steps for:

associating content transforms with a first and a second mobile client, the content transforms accounting for one or more operating characteristics of the first and second mobile clients;

producing first transformed content and second transformed content based on content from the content server and the content transforms; and

providing the first and second transformed content to the first and second mobile clients.

21. (Currently Amended) A method as recited in claim 20 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, ~~and-or~~ a cascaded mobile gateway.

22. (Original) A method as recited in claim 20 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising steps for:

producing a first additional transformed content based on a content transform associated with the first mobile client; and

providing the first additional transformed content to the first mobile client.

23. (Currently Amended) A method as recited in claim 20 wherein the one or more operating characteristics considered by the content transforms include at least one of the mobile clients' software, processor, memory, display, ~~and-or~~ communication link.

24. (Original) A method as recited in claim 20 wherein the computerized system includes a third mobile client, the method further comprising a step for providing the first transformed content to the third mobile client, due to similarities in one or more operating characteristics of the first and third mobile clients.

25. (Original) A method as recited in claim 20 wherein the content received from the content server is addressed to a list of mobile clients, the method further comprising a step for providing the content to each of the specific mobile clients contained in the list.

26. (Currently Amended) A method as recited in claim 20 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, ~~and~~ or sports data.



27. (Original) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a computer program product for implementing a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method at a mobile gateway; and

wherein said method is comprised of machine-executable instructions for performing the acts of:

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content to the first mobile client.

28. (Currently Amended) A computer program product as recited in claim 27 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, ~~and~~or communication link.

29. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of encrypting the content.

30. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of compressing the content.

31. (Original) A method as recited in claim 27 wherein the first transformed content comprises a notification that additional content is available at the content server, the method comprised further of machine-executable instructions for performing the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

32. (Currently Amended) A computer program product as recited in claim 27 wherein the first mobile client is one of a telephone, a pager, and a personal digital assistant and wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, ~~and~~ or sports data.

33. (Original) A computer program product as recited in claim 27 wherein the computerized system includes a second mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

34. (Original) A computer program product as recited in claim 33 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method comprised further of machine-executable instructions for performing the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

35. (Original) A computer program product as recited in claim 33 wherein the computerized system includes a third mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client;  
and

sending the first transformed content to the third mobile client.

## CLAIMS AFTER AMENDMENT B

1. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and a first and a second mobile client, the first and second mobile clients differing from each other in at least one operating characteristic, wherein the mobile gateway receives content that is addressed to the first and second mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client and assigning a second transform to the second mobile client, the first and second transforms specifically considering one or more operating characteristics of the first and second mobile clients;

receiving a list from the content server containing addresses for a plurality of mobile clients, including the first mobile client and the second mobile client;

receiving content from the content server, the content being addressed to the list;

altering the content according to the first and second transforms so that the content is compatible with the one or more operating characteristics of the first and second mobile clients, the altered content comprising a first transformed content and a second transformed content;

identifying an address for each mobile client contained within the list, including the first mobile client and the second mobile client;

addressing the first transformed content to the first mobile device and addressing the second transformed content to the second mobile device using the plurality of addresses received in the list;

establishing a communication link between the mobile gateway and the first and second mobile clients; and

sending the first transformed content to the first mobile client and sending the second transformed content to the second mobile client.

2. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms encrypting the content.

3. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms compressing the content.

4. (Previously Presented) A method as recited in claim 1 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

5. (Original) A method as recited in claim 1 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

6. (Previously Presented) A method as recited in claim 1 wherein the one or more operating characteristics considered by the first and second transforms include at least one of the first or second mobile client's software, processor, memory, display, or communication link.

7. (Original) A method as recited in claim 1 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

8. (Canceled).

9. (Previously Presented) A method as recited in claim 1 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.

10. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first configuration transform to the first mobile client, the first configuration transform customizing configuration information for one or more services available to the first mobile client;

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

determining that a change has occurred in at least one service available to the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

customizing configuration information relative to the change in the at least one service available to the first mobile client based on the first configuration transform to provide first transformed configuration information;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content and the first transformed configuration information to the first mobile client.

11. (Previously Presented) A method as recited in claim 10 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, or communication link.

12. (Original) A method as recited in claim 10 further comprising the act of the first transform encrypting the content.

13. (Original) A method as recited in claim 10 further comprising the act of the first transform compressing the content.

14. (Original) A method as recited in claim 10 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

15. (Previously Presented) A method as recited in claim 10 wherein the first mobile client is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

16. (Previously Presented) A method as recited in claim 10 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.



17. (Original) A method as recited in claim 10 wherein the computerized system includes a second mobile client, the method further comprising the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

18. (Original) A method as recited in claim 17 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

19. (Original) A method as recited in claim 17 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

20. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and mobile clients, wherein some of the mobile clients differ from each other in at least one operating characteristic, and wherein the mobile gateway receives content that is addressed to the mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising the mobile gateway performing steps for:

a step for associating content transforms with a first and a second mobile client, the content transforms accounting for one or more operating characteristics of the first and second mobile clients;

an act of receiving a list from the content server containing addresses for a plurality of mobile clients, including the first mobile client and the second mobile client;

a step for producing first transformed content and second transformed content based on content from the content server and the content transforms, the content received from the content server being addressed to the list; and

an act of addressing the first transformed content to the first mobile device and addressing the second transformed content to the second mobile device using the plurality of addresses received in the list; and

a step for providing the first and second transformed content to the first and second mobile clients.

21. (Previously Presented) A method as recited in claim 20 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

22. (Original) A method as recited in claim 20 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising steps for:

producing a first additional transformed content based on a content transform associated with the first mobile client; and

providing the first additional transformed content to the first mobile client.

23. (Previously Presented) A method as recited in claim 20 wherein the one or more operating characteristics considered by the content transforms include at least one of the mobile clients' software, processor, memory, display, or communication link.

24. (Original) A method as recited in claim 20 wherein the computerized system includes a third mobile client, the method further comprising a step for providing the first transformed content to the third mobile client, due to similarities in one or more operating characteristics of the first and third mobile clients.

25. (Canceled).

26. (Previously Presented) A method as recited in claim 20 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.

27. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a computer program product for implementing a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method at a mobile gateway; and

wherein said method is comprised of machine-executable instructions for performing the acts of:

assigning a first configuration transform to the first mobile client, the first configuration transform customizing configuration information for one or more services available to the first mobile client;

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

determining that a change has occurred in at least one service available to the first mobile client;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

customizing configuration information relative to the change in the at least one service available to the first mobile client based on the first configuration transform to provide first transformed configuration information;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content and the first transformed configuration information to the first mobile client.

28. (Previously Presented) A computer program product as recited in claim 27 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, or communication link.

29. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of encrypting the content.

30. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of compressing the content.

31. (Original) A method as recited in claim 27 wherein the first transformed content comprises a notification that additional content is available at the content server, the method comprised further of machine-executable instructions for performing the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

32. (Previously Presented) A computer program product as recited in claim 27 wherein the first mobile client is one of a telephone, a pager, and a personal digital assistant and wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.

33. (Original) A computer program product as recited in claim 27 wherein the computerized system includes a second mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

34. (Original) A computer program product as recited in claim 33 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method comprised further of machine-executable instructions for performing the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

35. (Original) A computer program product as recited in claim 33 wherein the computerized system includes a third mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and sending the first transformed content to the third mobile client.

## CLAIMS AFTER AMENDMENT C

1. (Currently Amended) At a mobile gateway ~~in~~ a computerized system that includes a content server, ~~a~~the mobile gateway, and a first and a second mobile client, the first and second mobile clients differing from each other in at least one operating characteristic, wherein the mobile gateway receives content that is addressed to the first and second mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client and assigning a second transform to the second mobile client, the first and second transforms specifically considering one or more operating characteristics of the first and second mobile clients;

receiving a list from the content server containing addresses for a plurality of mobile clients, including the first mobile client and the second mobile client;

receiving content from the content server, the content being addressed to the list, wherein the content has not yet been altered in accordance with the first or second transform;

identifying from one or more of the received content and the received list that the first transform and the second transform are to be applied;

altering the content according to the first and second transforms so that the content is compatible with the one or more operating characteristics of the first and second mobile clients, the altered content comprising a first transformed content and a second transformed content;

identifying an address for each mobile client contained within the list, including the first mobile client and the second mobile client;

addressing the first transformed content to the first mobile device and addressing the second transformed content to the second mobile device using the plurality of addresses received in the list;

establishing a communication link between the mobile gateway and the first and second mobile clients; and

sending the first transformed content to the first mobile client and sending the second transformed content to the second mobile client.

2. (Original) A method as recited in claim 0 further comprising the act of at least one of the transforms encrypting the content.

3. (Original) A method as recited in claim 0 further comprising the act of at least one of the transforms compressing the content.

4. (Previously Presented) A method as recited in claim 0 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

5. (Currently Amended) A method as recited in claim 0 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server, wherein the additional content has not been altered in accordance with any one or more of the first or second transforms;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

6. (Previously Presented) A method as recited in claim 0 wherein the one or more operating characteristics considered by the first and second transforms include at least one of the first or second mobile client's software, processor, memory, display, or communication link.



7. (Original) A method as recited in claim 0 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client;  
and

sending the first transformed content to the third mobile client.

8. (Canceled).

9. (Previously Presented) A method as recited in claim 0 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.

10. (Currently Amended) At a mobile gateway in a computerized system that includes a content server, ~~a~~the mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

~~assigning a first configuration transform to the first mobile client, the first configuration transform customizing configuration information for one or more services available to the first mobile client;~~

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

determining that a change has occurred in at least one service available to the first mobile client, such that prior hardware or software configuration information of the first mobile client is incompatible with a current version of the at least one service;

creating first transformed configuration information at the mobile gateway, wherein the first transformed configuration information is consistent with the change in the at least one service;

receiving content from the content server;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client and the change to the at least one service, the altered content comprising a first transformed content;

~~customizing configuration information relative to the change in the at least one service available to the first mobile client based on the first configuration transform to provide first transformed configuration information;~~

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content and the first transformed configuration information to the first mobile client.

11. (Previously Presented) A method as recited in claim 10 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, or communication link.

12. (Original) A method as recited in claim 10 further comprising the act of the first transform encrypting the content.

13. (Original) A method as recited in claim 10 further comprising the act of the first transform compressing the content.

14. (Original) A method as recited in claim 10 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

15. (Previously Presented) A method as recited in claim 10 wherein the first mobile client is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

16. (Previously Presented) A method as recited in claim 10 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.

17. (Original) A method as recited in claim 10 wherein the computerized system includes a second mobile client, the method further comprising the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

18. (Original) A method as recited in claim 17 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

19. (Original) A method as recited in claim 17 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

20. (Currently Amended) ~~In~~At a mobile gateway in a computerized system that includes a content server, ~~a~~the mobile gateway, and mobile clients, wherein some of the mobile clients differ from each other in at least one operating characteristic, and wherein the mobile gateway receives content that is addressed to the mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising the mobile gateway performing:

- a step for associating content transforms with a first and a second mobile client, the content transforms accounting for one or more operating characteristics of the first and second mobile clients;

- an act of receiving a list from the content server containing addresses for a plurality of mobile clients, including the first mobile client and the second mobile client

- an act of the mobile gateway identifying at least from the received list that the content transforms for the first and second mobile clients are to be used;

- a step for producing first transformed content and second transformed content based on content from the content server and the content transforms, the content received from the content server being addressed to the list;

- an act of addressing the first transformed content to the first mobile device and addressing the second transformed content to the second mobile device using the plurality of addresses received in the list; and

- a step for providing the first and second transformed content to the first and second mobile clients.

21. (Previously Presented) A method as recited in claim 20 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

22. (Original) A method as recited in claim 20 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising steps for:

producing a first additional transformed content based on a content transform associated with the first mobile client; and

providing the first additional transformed content to the first mobile client.

23. (Previously Presented) A method as recited in claim 20 wherein the one or more operating characteristics considered by the content transforms include at least one of the mobile clients' software, processor, memory, display, or communication link.

24. (Original) A method as recited in claim 20 wherein the computerized system includes a third mobile client, the method further comprising a step for providing the first transformed content to the third mobile client, due to similarities in one or more operating characteristics of the first and third mobile clients.

25. (Canceled).

26. (Previously Presented) A method as recited in claim 20 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.

27. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a computer program product for implementing a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method at a mobile gateway; and

wherein said method is comprised of machine-executable instructions for performing the acts of:

assigning a first configuration transform to the first mobile client, the first configuration transform customizing configuration information for one or more services available to the first mobile client;

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

determining that a change has occurred in at least one service available to the first mobile client;

receiving content from the content server, wherein the content has not yet been altered in accordance with the first or second transform;

identifying from one or more of the received content and the received list that the first transform and the second transform are to be applied;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

customizing configuration information relative to the change in the at least one service available to the first mobile client based on the first configuration transform to provide first transformed configuration information;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content and the first transformed configuration information to the first mobile client.

28. (Previously Presented) A computer program product as recited in claim 27 wherein the one or more operating characteristics considered by the first transform include at least one of the first mobile client's software, processor, memory, display, or communication link.

29. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of encrypting the content.

30. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of compressing the content.

31. (Original) A method as recited in claim 27 wherein the first transformed content comprises a notification that additional content is available at the content server, the method comprised further of machine-executable instructions for performing the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server, wherein the additional content has not been altered in accordance with any one or more of the first or second transforms;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.



32. (Previously Presented) A computer program product as recited in claim 27 wherein the first mobile client is one of a telephone, a pager, and a personal digital assistant and wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.

33. (Original) A computer program product as recited in claim 27 wherein the computerized system includes a second mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

34. (Original) A computer program product as recited in claim 33 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method comprised further of machine-executable instructions for performing the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

35. (Original) A computer program product as recited in claim 33 wherein the computerized system includes a third mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and sending the first transformed content to the third mobile client.

## CLAIMS AFTER AMENDMENT D

1. (Currently Amended) At a mobile gateway in a computerized system that includes a content server, the mobile gateway, and a first and a second mobile client, the first and second mobile clients differing from each other in at least one operating characteristic, wherein the mobile gateway receives content that is addressed to the first and second mobile clients from the content server, a method of customizing the content based on at least one operating characteristic of each mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the first mobile client and assigning a second transform to the second mobile client, the first and second transforms specifically considering one or more operating characteristics of the first and second mobile clients;

receiving a list from the content server containing addresses for a plurality of mobile clients, including the first mobile client and the second mobile client;

receiving content from the content server, the content being addressed to the list, wherein the content has not yet been altered in accordance with the first or second transform;

~~identifying from one or more of the received content and the received list~~  
determining at the mobile gateway that the first transform and the second transform are to be applied to the received content upon the mobile gateway identifying that the list includes an address for the first mobile client and an address for the second mobile client;

altering the content at the mobile gateway according to the first and second transforms so that the content is compatible with the one or more operating characteristics of the first and second mobile clients, the altered content comprising a first transformed content and a second transformed content;

~~identifying an address for each mobile client contained within the list, including the first mobile client and the second mobile client;~~

addressing the first transformed content to the first mobile device and addressing the second transformed content to the second mobile device using the plurality of addresses received in the list;

establishing a communication link between the mobile gateway and the first and second mobile clients; and

sending the first transformed content to the first mobile client and sending the second transformed content to the second mobile client.

2. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms encrypting the content.

3. (Original) A method as recited in claim 1 further comprising the act of at least one of the transforms compressing the content.

4. (Previously Presented) A method as recited in claim 1 wherein at least one of the mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

5. (Previously Presented) A method as recited in claim 1 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server, wherein the additional content has not been altered in accordance with any one or more of the first or second transforms;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

6. (Currently Amended) A method as recited in claim 1 wherein the one or more operating characteristics considered by the first and second transforms include the first and second mobile client's memory capabilities, and at least one of the first or second mobile client's software, processor, ~~memory~~, display, or communication link.

7. (Original) A method as recited in claim 1 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client;  
and

sending the first transformed content to the third mobile client.

8. (Canceled).

9. (Previously Presented) A method as recited in claim 1 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.

10. (Currently Amended) At a mobile gateway in a computerized system that includes a content server, the mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, the method comprising a mobile gateway performing the acts of:

assigning a first transform to the a first mobile client and a second transform to a second mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

determining at the mobile gateway that a change has occurred ~~in at least one service available to with the~~ one or more operating characteristics of the first mobile client, such that prior hardware or software configuration information of the first mobile client is incompatible with ~~a current version of the at least one service~~ the first transform;

creating ~~first transformed configuration information~~ an updated first transform at the mobile gateway, wherein the updated first transformed configuration information is consistent with the change in ~~the at least one service~~ operating characteristics of the first mobile client;

receiving content from the content server;

altering the content at the mobile gateway according to the updated first transform and with the second transform so that the content is compatible with the change in the one or more operating characteristics of the first mobile client and with the one or more operating characteristics of the second mobile client ~~and the change to the at least one service, the altered content comprising a first transformed content~~;

establishing a communication link between the mobile gateway and the first mobile client; and

the mobile gateway sending the content altered in accordance with the updated first transformed content and the first transformed configuration information to the first

mobile client, and sending the content altered in accordance with the second transform to the second mobile client.

11. (Currently Amended) A method as recited in claim 10 wherein the one or more operating characteristics considered by the updated first transform include the first mobile client's memory capabilities, and at least one of the first mobile client's software, processor, ~~memory~~, display, or communication link.

12. (Original) A method as recited in claim 10 further comprising the act of the first transform encrypting the content.

13. (Original) A method as recited in claim 10 further comprising the act of the first transform compressing the content.

14. (Currently Amended) A method as recited in claim 10 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising the acts of:

receiving a request for the additional content from the first mobile client;

retrieving the additional content from the content server;

altering the additional content according to the updated first transform so that the content is compatible with the change in one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

15. (Previously Presented) A method as recited in claim 10 wherein the first mobile client is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

16. (Previously Presented) A method as recited in claim 10 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports content.

17. (Currently Amended) A method as recited in claim 10 ~~wherein the computerized system includes a second mobile client, the method~~ further comprising the acts of:

~~assigning~~ updating the second transform to the second mobile client, the updated second transform specifically considering a change in one or more operating characteristics of the second mobile client;

altering the content according to the updated second transform at the mobile gateway so that the content received from the content server is compatible with the change in one or more operating characteristics of the second mobile client, ~~the altered content comprising a second transformed content;~~

establishing a communication link between the mobile gateway and the second mobile client; and

sending the content altered according to the updated second transformed content to the second mobile client.

18. (Original) A method as recited in claim 17 wherein the content received from the content server is addressed to a list containing the first and second mobile clients, the method further comprising the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

19. (Original) A method as recited in claim 17 wherein the computerized system includes a third mobile client, the method further comprising the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and

sending the first transformed content to the third mobile client.

20. (Cancelled).

21. (Currently Amended) A method as recited in claim 2036 wherein at least one of the first or second mobile clients is one of a telephone, a pager, a personal digital assistant, or a cascaded mobile gateway.

22. (Currently Amended) A method as recited in claim 2036 wherein the first transformed content comprises a notification that additional content is available at the content server, the method further comprising steps for:

producing a first additional transformed content based on a content transform associated with the first mobile client; and

providing the first additional transformed content to the first mobile client.

23. (Currently Amended) A method as recited in claim 2036 wherein the one or more operating characteristics considered by the content transforms include the mobile client's memory capabilities, and at least one of the mobile clients' software, processor, ~~memory~~, display, or communication link.

24. (Currently Amended) A method as recited in claim 2036 wherein the computerized system includes a third mobile client, the method further comprising a step for providing content transformed by an updated first content transform ~~the first transformed content~~ to the third mobile client, due to similarities in one or more operating characteristics of the first and third mobile clients.

25. (Canceled).

26. (Currently Amended) A method as recited in claim 2036 wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.



27. (Currently Amended) In a computerized system that includes a content server, a mobile gateway, and a first mobile client, wherein the mobile gateway receives from the content server, content that is addressed to the first mobile client, a computer program product for implementing a method of customizing the content based on at least one operating characteristic of the first mobile client, wherein the customizing avoids further processing at the content server, comprising:

a computer readable medium for carrying machine-executable instructions for implementing the method at a mobile gateway; and

wherein said method is comprised of machine-executable instructions for performing the acts of:

assigning a first configuration transform to the first mobile client, the first configuration transform customizing configuration information for one or more services available to the first mobile client;

assigning a first transform to the first mobile client, the first transform specifically considering one or more operating characteristics of the first mobile client;

determining that a change has occurred in at least one service available to the first mobile client;

receiving content and a recipient list for the content from the content server, wherein the content has not yet been altered in accordance with the first or second transform;

identifying from ~~one or more of the received content and~~ the received recipient list for the content that the first transform and the second transform are to be applied;

altering the content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the altered content comprising a first transformed content;

customizing configuration information relative to the change in the at least one service available to the first mobile client based on the first configuration transform to provide first transformed configuration information;

establishing a communication link between the mobile gateway and the first mobile client; and

sending the first transformed content and the first transformed configuration information to the first mobile client.

28. (Currently Amended) A computer program product as recited in claim 27 wherein the one or more operating characteristics considered by the first transform include the first mobile client's memory at least one of the first mobile client's software, processor, ~~memory~~, display, or communication link.

29. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of encrypting the content.

30. (Original) A computer program product as recited in claim 27, the method comprised further of machine-executable instructions for the first transform performing the act of compressing the content.

31. (Original) A method as recited in claim 27 wherein the first transformed content comprises a notification that additional content is available at the content server, the method comprised further of machine-executable instructions for performing the acts of:

receiving a request for the additional content from the first mobile client,

retrieving the additional content from the content server, wherein the additional content has not been altered in accordance with any one or more of the first or second transforms;

altering the additional content according to the first transform so that the content is compatible with the one or more operating characteristics of the first mobile client, the act of altering producing a first transformed additional content; and

sending the first transformed additional content to the first mobile client.

32. (Previously Presented) A computer program product as recited in claim 27 wherein the first mobile client is one of a telephone, a pager, and a personal digital assistant and wherein the content comprises one of email, calendar, contact, task, Web, notification, financial, configuration, or sports data.

33. (Original) A computer program product as recited in claim 27 wherein the computerized system includes a second mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning a second transform to the second mobile client, the second transform specifically considering one or more operating characteristics of the second mobile client;

altering the content according to the second transform so that the content is compatible with the one or more operating characteristics of the second mobile client, the altered content comprising a second transformed content;

establishing a communication link between the mobile gateway and the second mobile client; and

sending the second transformed content to the second mobile client.

34. (Original) A computer program product as recited in claim 33 wherein the content received from the content server is addressed to a list containing the first and second

mobile clients, the method comprised further of machine-executable instructions for performing the act of addressing the content specifically to the first mobile client and to the second mobile client as defined in the list.

35. (Original) A computer program product as recited in claim 33 wherein the computerized system includes a third mobile client, the method comprised further of machine-executable instructions for performing the acts of:

assigning the first transform to the third mobile client, the first transform specifically considering one or more operating characteristics of the third mobile client; and sending the first transformed content to the third mobile client.

36. (New) At a mobile gateway in a computerized system in which the mobile gateway receives content intended to be pushed to mobile clients from a content server, a method of the mobile gateway customizing the content from the content server based one or more operating characteristics identified for a plurality of intended recipients, comprising the acts of:

receiving content from a content server at the mobile gateway, wherein the content is addressed by the content server to at least a first mobile client, a second mobile client, and a different mobile gateway;

a step for determining at the mobile gateway an appropriate content transform for each of the first mobile client, the second mobile client, and for the different mobile gateway based on detected operating characteristics for each of the first mobile client, the second mobile client, and for the different mobile gateway;

assigning at the mobile gateway a first content transform to the first mobile client, a second content transform to the second mobile client, and a third content transform to the different mobile gateway, wherein the first, second, and third content transforms correspond to the detected one or more operating characteristics of the first mobile client, the second mobile client, and the devices managed by the different mobile gateway;

the mobile gateway sending content transformed by an updated first content transform to the first mobile client, content transformed by the second content transform to the second mobile client, and content transformed by the third content transform to the different mobile gateway.

37. (New) A method as recited in claim 36, wherein the step for determining an appropriate transform comprises the acts of:

identifying one or more operating characteristics for each of the first mobile client and the second mobile client;

identifying one or more operating characteristics for devices managed by the different mobile gateway; and

updating the first transform upon identifying a change in memory capacity of the first mobile client.

38. (New) A method as recited in claim 37, further comprising the acts of:

receiving new content from the content server, wherein at least a portion of the new content is too large for the first mobile client based on the change in memory capacity of the first mobile client;

transforming the new content in accordance with the second content transform and in accordance with the third content transform; and

sending the transformed new content to the second mobile client, and the different mobile gateway, but not to the first mobile client.

## **EXHIBIT B**

General

Summary

Statistics

Contents

Custom

Created: Monday, October 20, 1997 5:12:14 PM

Modified: Friday, June 02, 2006 11:38:14 AM

Accessed: Tuesday, June 13, 2006 4:53:53 PM

Printed: Monday, April 06, 1998 11:37:52 AM

Last saved by: Marc Seinfeld

Revision number: 203

Total editing time: 5518 Minutes

Statistics:

Statistic name	Value
Slides:	24
Paragraphs:	287
Words:	1316
Bytes:	338561
Notes:	15
Hidden slides:	0
Multimedia clips:	0
Presentation format:	On-screen Show



# Exchange Mobility

Marc Seinfeld

Lead Program Manager

Microsoft Exchange Server Mobile Connectivity

Microsoft Confidential

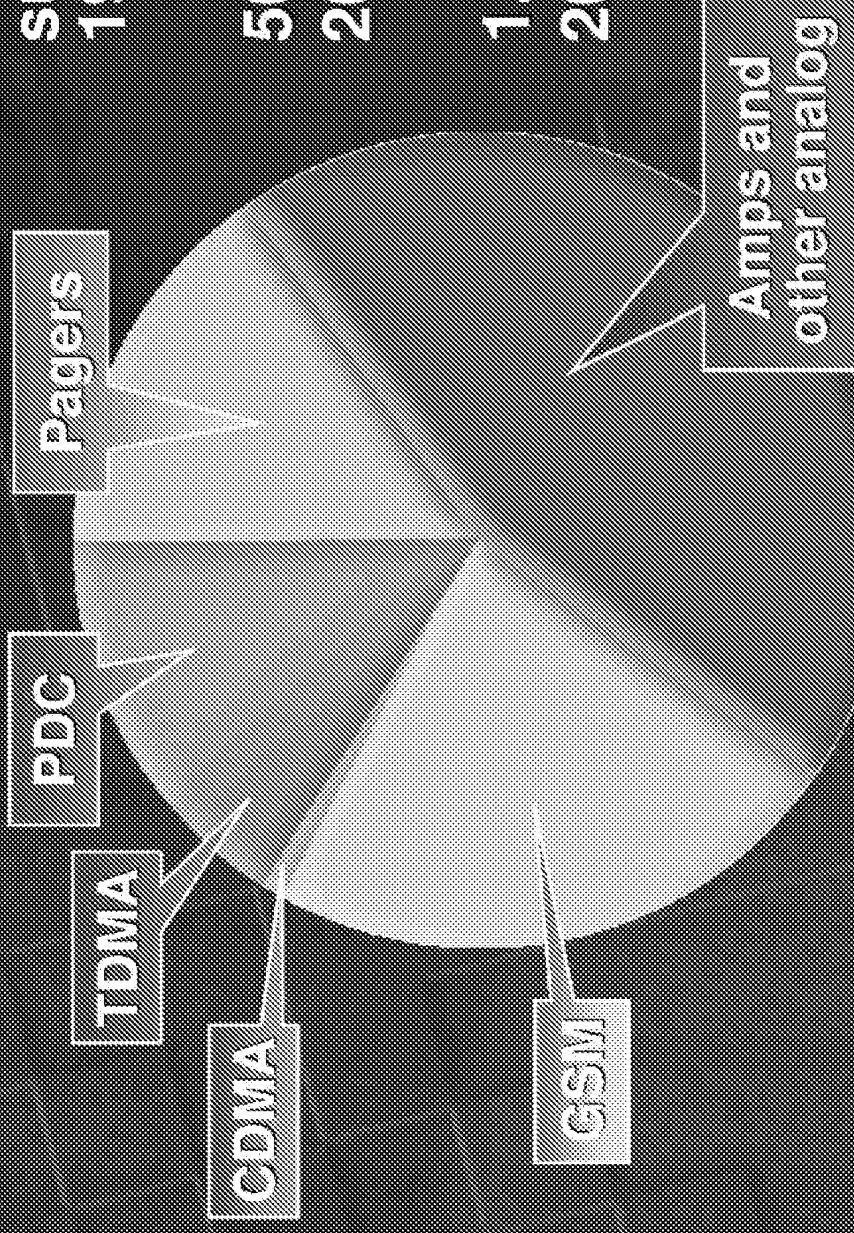
# Why Wireless Now?

- ◆ Wireless communications market is large and rapidly growing

198 million  
world-wide  
subscribers in  
1997

500 million by  
2002

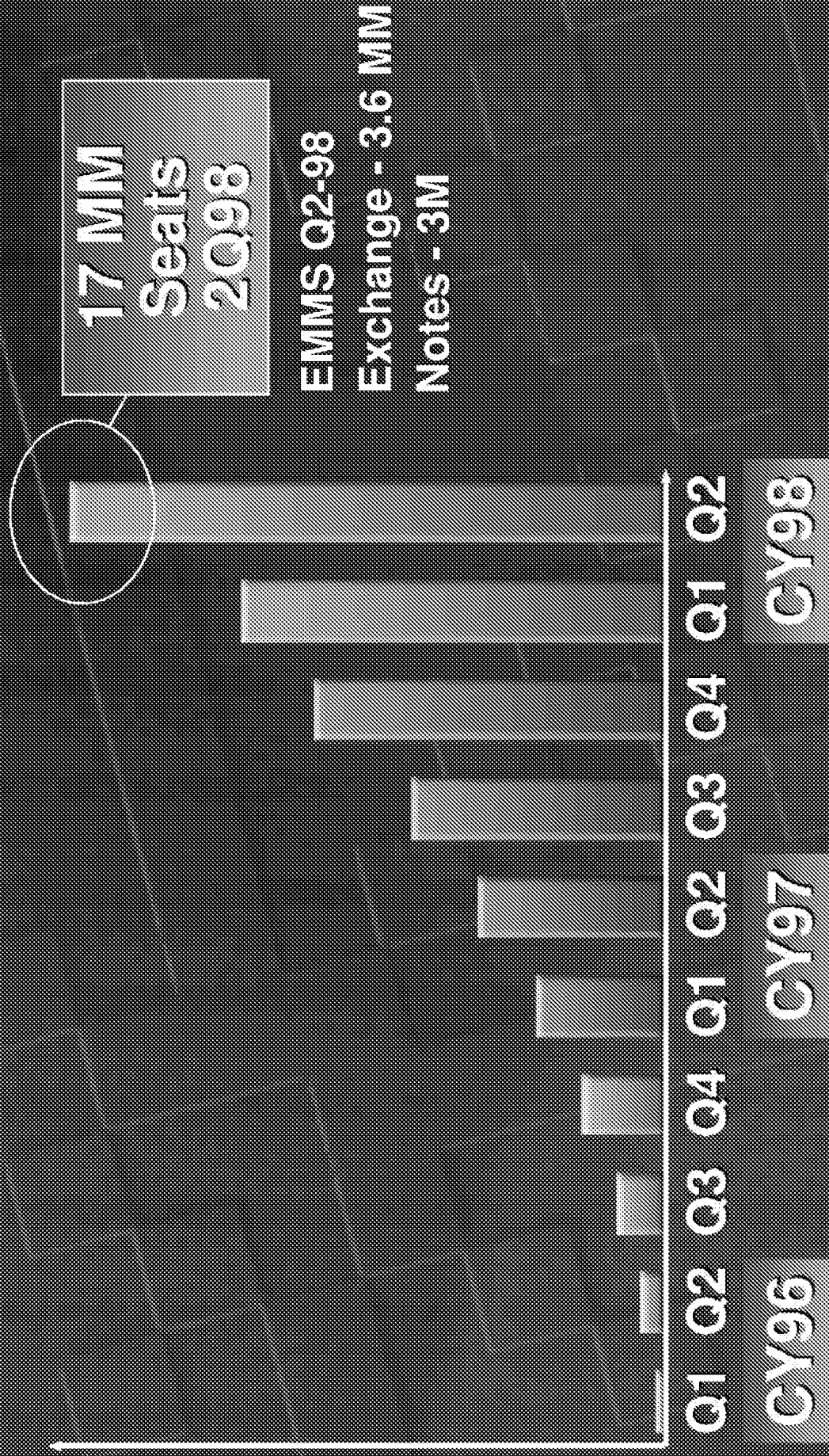
1.1 billion by  
2005





# Exchange after 2¼ Years

Fastest growing server product  
in PC history



# Exchange Server

Q1 1996

Exchange 4.0

Client/Server  
Foundation

Q1 1997

Exchange 5.0

PIM,  
Collaboration  
Internet

Q4 1997

Exchange 5.5

Scalability  
Three-tier  
applications

Exchange, NT 5.0

The Next  
Generation

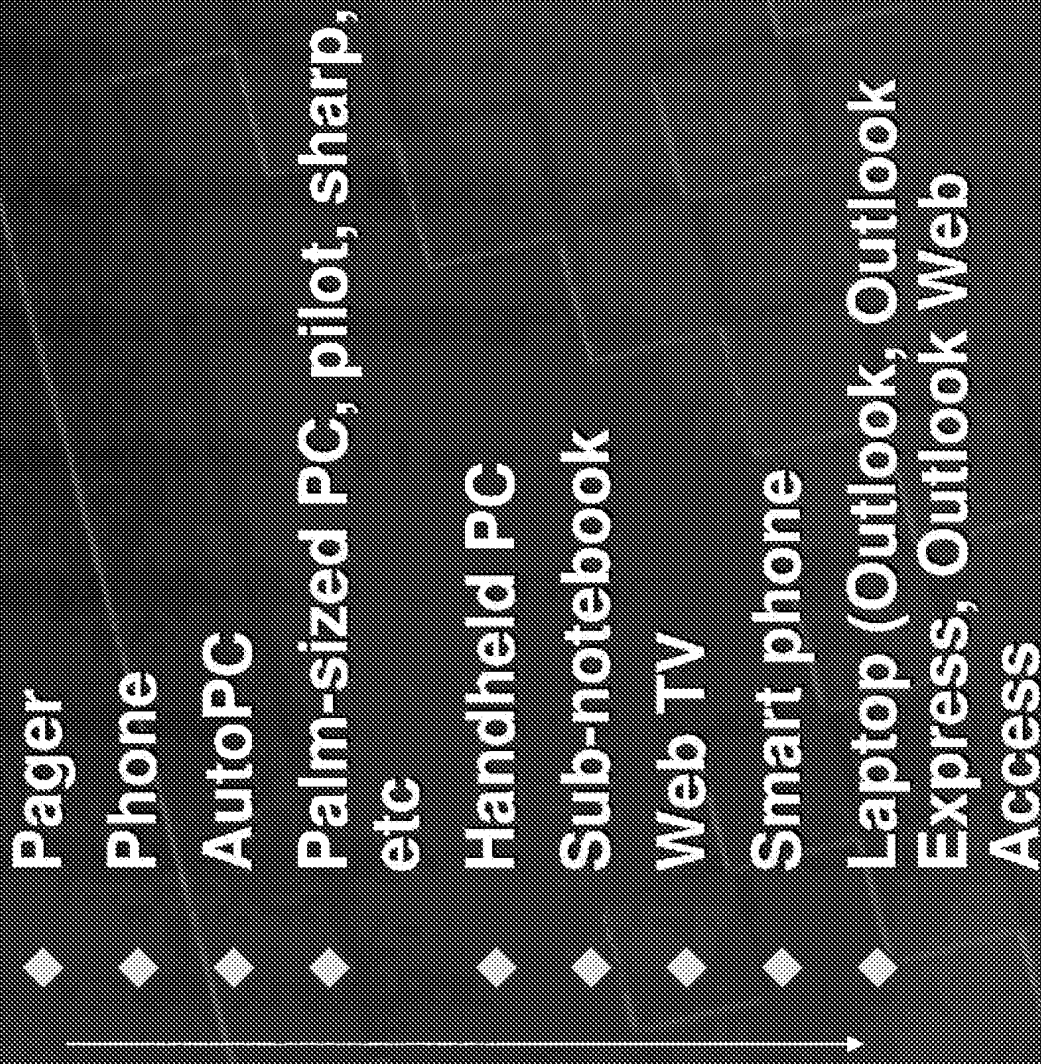
Microsoft Confidential



# **Exchange Platinum: A Platform for Mobility**

- ◆ **Active directory integration**
- ◆ **Multi-tiered management infrastructure**
- ◆ **Internet content and protocols**
- ◆ **Common data access via OLE DB**
- ◆ **Integrated DNS based collaboration**
- ◆ **A platform for enhanced services**

# Spectrum of Wireless Devices (Potential Exchange Clients)

- 
- ◆ Pager
  - ◆ Phone
  - ◆ AutoPC
  - ◆ Palm-sized PC, pilot, sharp, etc
  - ◆ Handheld PC
  - ◆ Sub-notebook
  - ◆ Web TV
  - ◆ Smart phone
  - ◆ Laptop (Outlook, Outlook Express, Outlook Web Access)



## **Carriers are anxious to offer end-to-end service**

- ◆ **User's own mail, calendar, contacts**
  - Hosted Exchange and tunneling support into corporate Exchange data
  - One mailbox; one address - not a separate one for my mobile device
- ◆ **Over the air service provisioning and billing**
  - Activate/subscribe/modify services, review service information (minutes left)
- ◆ **General information like stock quotes, traffic, weather**

# Goals

- ◆ Access to Exchange ,other BackOffice and other server/web data
  - A *single mailbox*, rather than a separate mailbox/address for my wireless account
- ◆ Access from a variety of mobile devices, from pagers and mobile phones to smarter devices
  - Eventually full IP packet, but in the interim ‘prime the pump’ via exploiting SMS and proving demand for wireless data while selling carrier minutes



# Scenarios

**Access to Exchange mail, calendar, contacts, and tasks from wireless devices (pager, phone, CE)**

- **Push/trickle data down to device**
  - **continuous (passive) sync also possible w/smart device**
- **3-line or richer fidelity browsing**
  - **WAP & WAP-like alternatives**
  - **2 way message based, non-IP transport (ie. SMS, paging, Mobitex, etc.)**

# Scenarios (Continued)

Synchronize mail, calendar, contacts, and tasks *directly* with Exchange Server

- Smart device sync on demand
- Outlook's mobile access



# Technology Goals

## ◆ Customer requirement

- Providing the enabling infrastructure for a broad range of mobile devices to interoperate with Exchange and other BackOffice servers
- Provide framework for access to generic data on the web via a wireless device
- Increase the value of mobile wireless devices and gain new users of Exchange

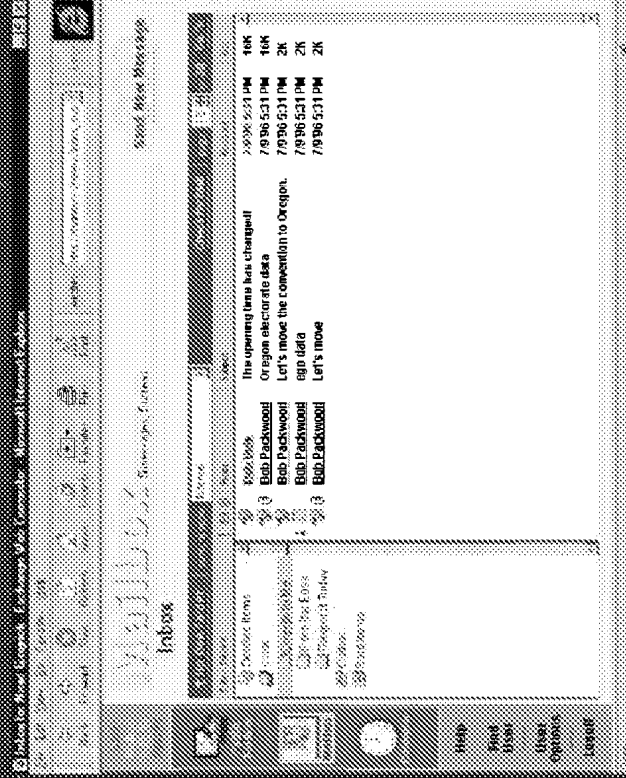
## ◆ Resulting Goals - Establish Pt as a Mobility Platform providing:

- Registration - register new devices, their capabilities and presence
- Personalization - subscribe to specific desired information
- Access - manipulate / reformat information for device
- Network / Transport - bind to specific transports and address performance, latency and security.

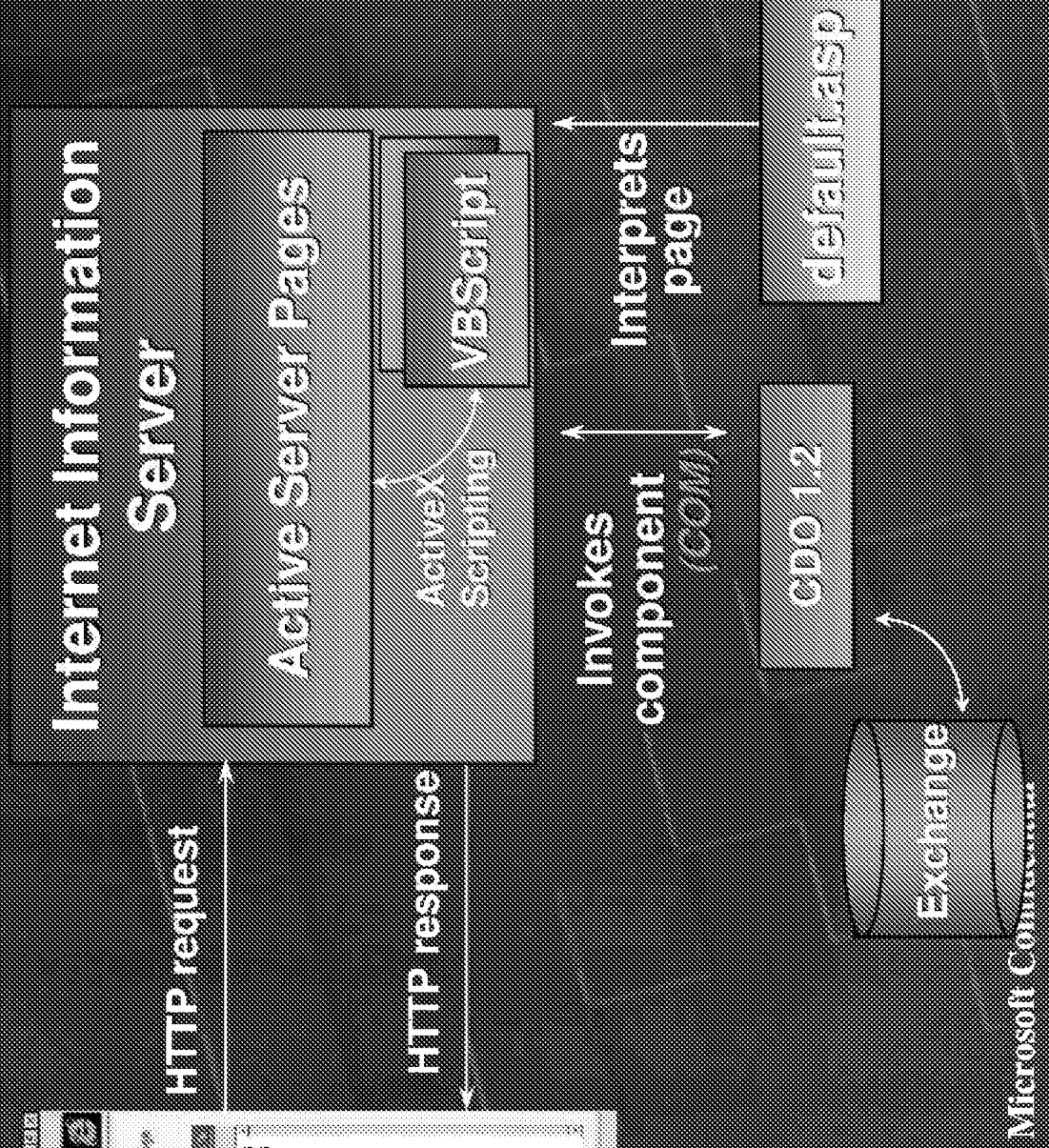
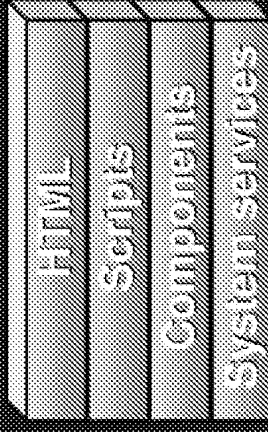
# Active Server Pages

## Application flow

Client

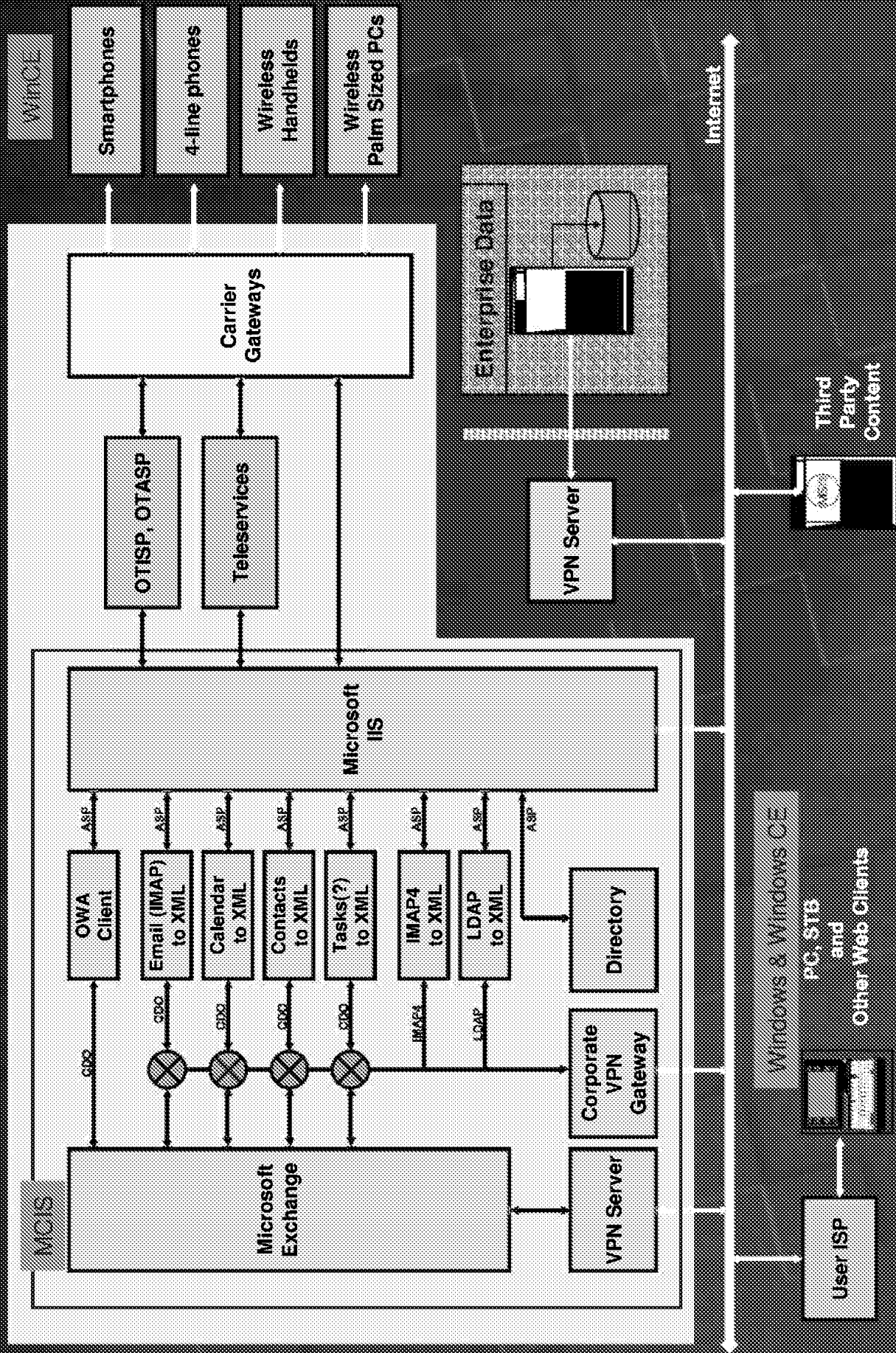


Server

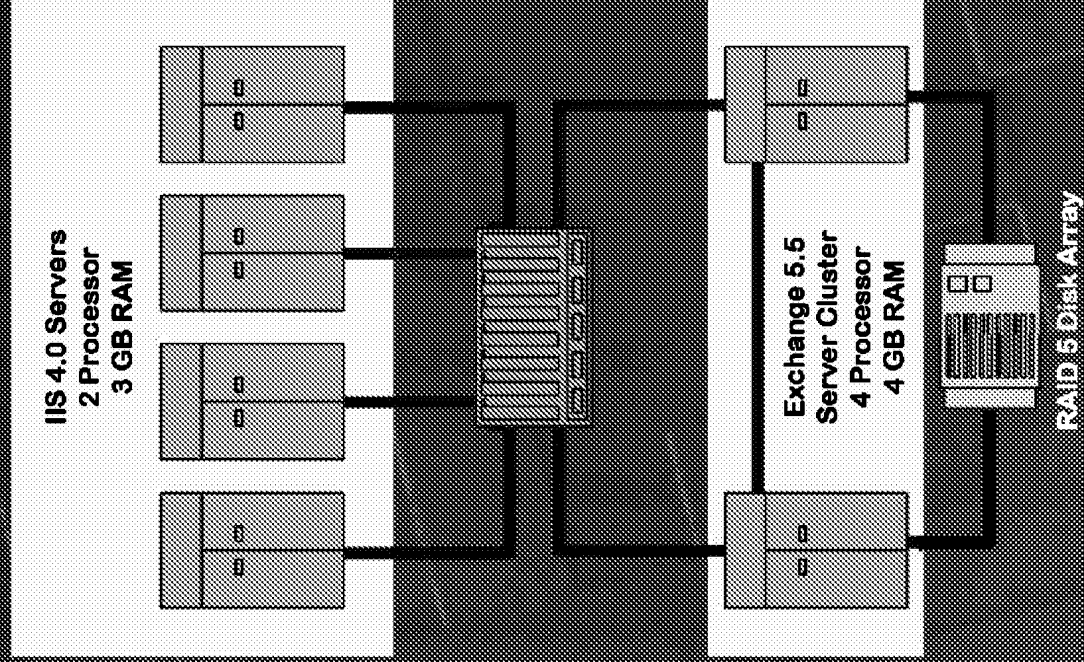




# End-to-End Architecture Overview



# NOC Networking (25K User Support)





# DMI Infrastructure

- Proxies http over message-based (SMS) Xport in absence of IP
- Modular design allowing for optimization across:
  - various devices
  - various networks
  - various physical connections
- ‘Sync-aware’ for benefit of smart devices (v. today’s SMTP mail to device)
- use DMI when your network’s not fast, fat and free

```

graph TD
    Gateway[Gateway Module] --> Device[Device Module]
    Device --> Carrier[Carrier Module]
    Carrier --> Protocol[Protocol Module]
    Protocol --> Locator[Locator]

```



# Sync work objectives

- ◆ Remove Sync dependency on desktop and Outlook Object Model and sync directly with the server
- ◆ Provide Server side sync efficiencies and optimizations (object not session based)
- ◆ Provide common sync interface to a variety of data stores
- ◆ Minimize CE device-side code
- ◆ Implement modular design to reduce memory footprint

## **Prototypes**

- ◆ **Proof of concept for Platinum Device Mobility Interconnect, built on Ex5.5**
  - Internal and external sales tool to demonstrate DMI functionality with a variety of common devices
  - Acts as sniffer for the system
  - Drives requirements on Exchange components

## **◆ Components**

- Calendar, Contact, Task trickle updates
- SMTP Forwarder
- CE Pager Client
- Browse and/or message response agent
- Information Request Handler
- CEMail client ASP built on OWA



# BACKUP INFO

# Demo

- ◆ **Trickle updates to PIM objects**

Ericl is attending meetings around campus and away from his office... concurrently lindahi makes updates to his schedule and a related contact... ericl receives the updates.

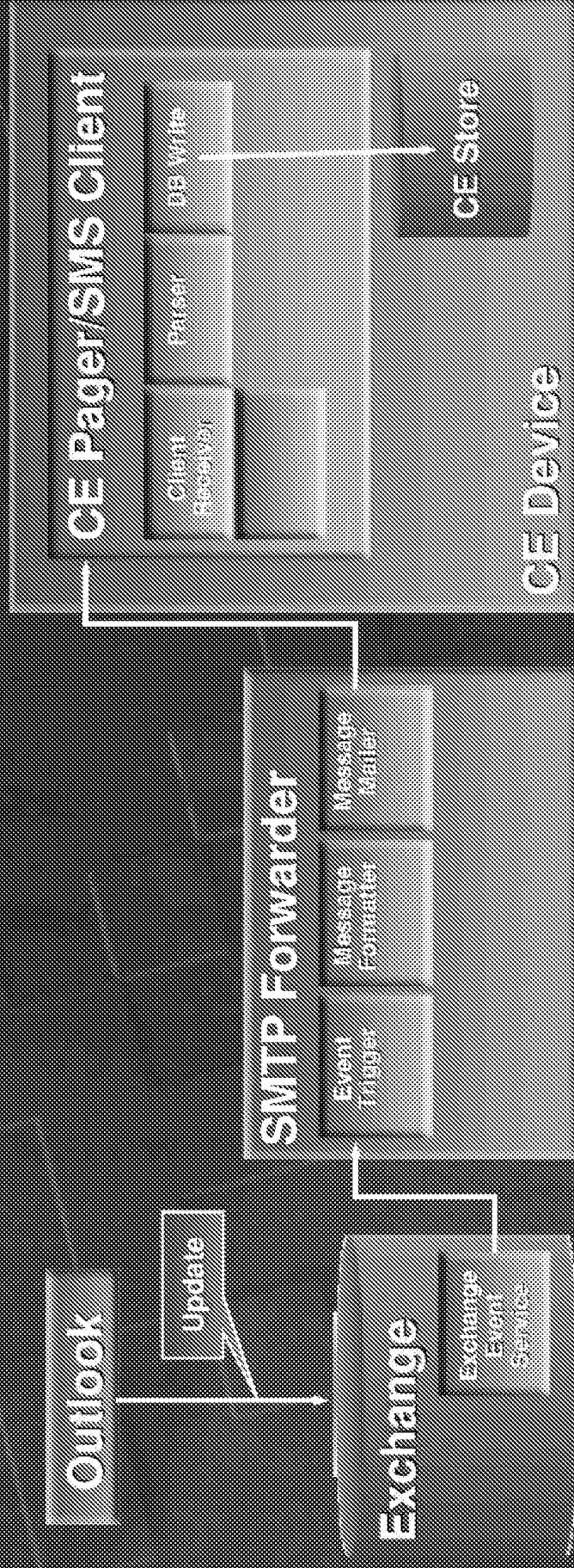
- ◆ **Information Request Agent**

On route to a mtg. with markled, ericl queries for the bldg/rm number as it's not included in the meeting request. Later that day, ericl's driving home and wants to check the closing MSFT price, and get the 5 day forecast for a sailing excursion he's planning



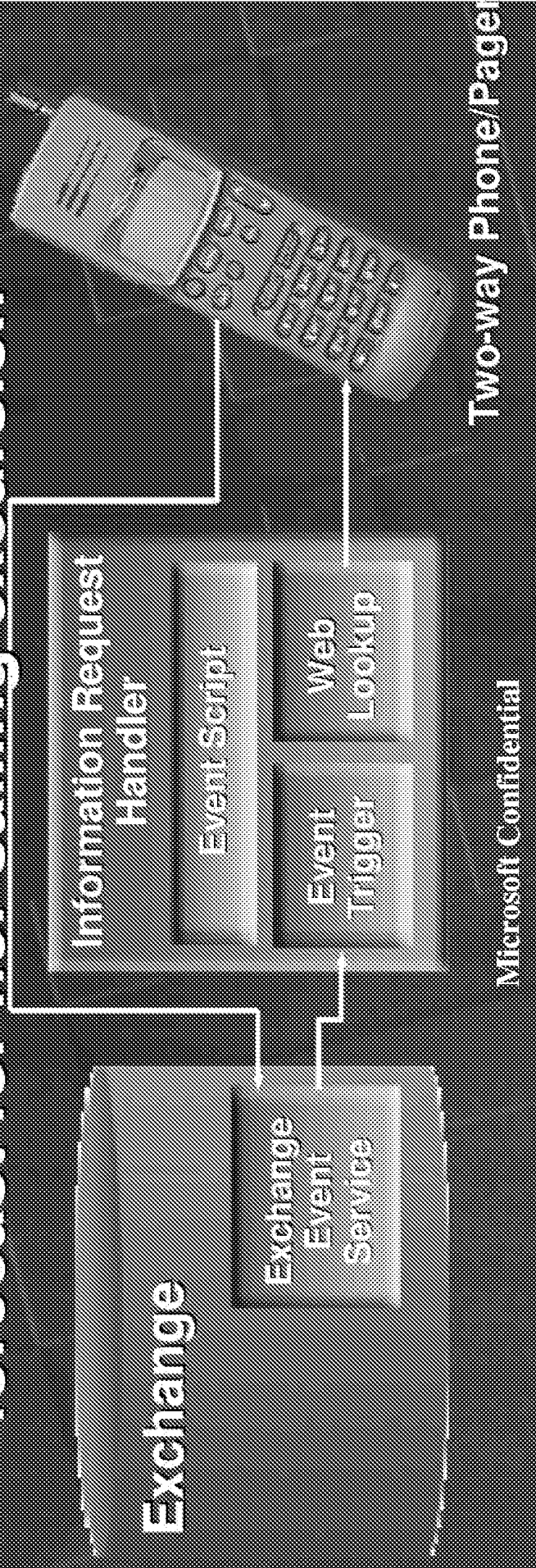
# Continuous PIM Updates

- ◆ While Susan presents a session, her assistant updates her meeting schedule and adds a related contact...
- ◆ Susan receives the updates



# Information Request Agent

- ◆ While out of the office, Susan looks up Neil's office number for her next meeting
- ◆ While driving home, she checks the closing MSFT price and gets a five-day forecast for her sailing excursion



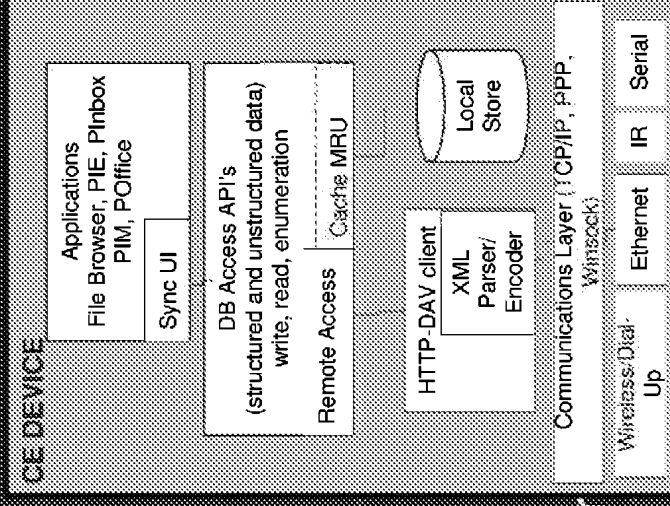


# Phone Micro-browsers

- ◆ See <http://www.attws.com/nohost/data/pocketnet/pn.html>
- ◆ For a richer (yet simple browser based) experience - CEMail .ASP scripts
  - Trivial to optimize these scripts for alternate form factors
  - Can be found on <http://www.microsoft.com/exchange/appfarm>

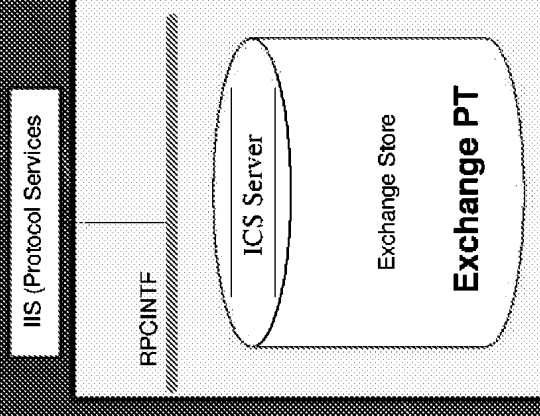
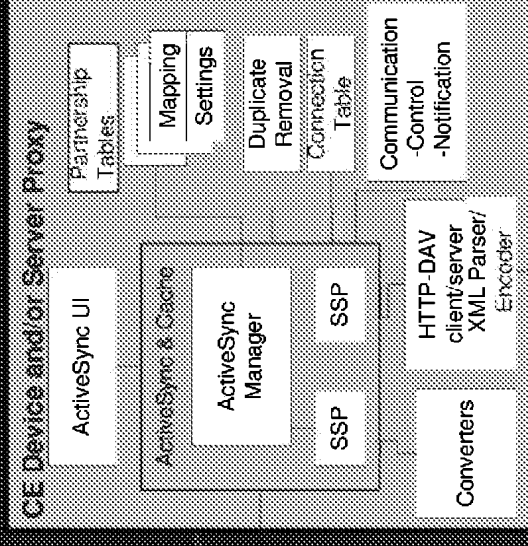
# CE/Exchange Synchronization

- ◆ **CE Device or Server Proxy (CE)**
  - **Status Monitoring**
  - **Sync Initiation**
  - **Enumeration**
  - **Determine & control changes/deletes**
  - **Conflict Management**
  - **Combine / Discard**
  - **Document Conversion**
  - **Profile Management**



## ◆ HTTP-DAV

- **-Change Identification, Tracking & Notification**
- **- Object Identification, Enumeration, Serialization**
- **- Create/Modify/Delete Object**
- **- Object & Property Filtering**





## **EXHIBIT C**

General

Summary

Statistics

Contents

Custom

Created: Sunday, April 05, 1998 10:58:00 PM  
Modified: Friday, June 02, 2006 11:38:13 AM  
Accessed: Tuesday, June 13, 2006 4:56:01 PM  
Printed:

Last saved by: Neil Fishman  
Revision number: 5  
Total editing time: 190 Minutes

Statistics:

Statistic name	Value
Pages:	2
Paragraphs:	23
Lines:	96
Words:	780
Characters:	4008
Characters (with spaces):	4785

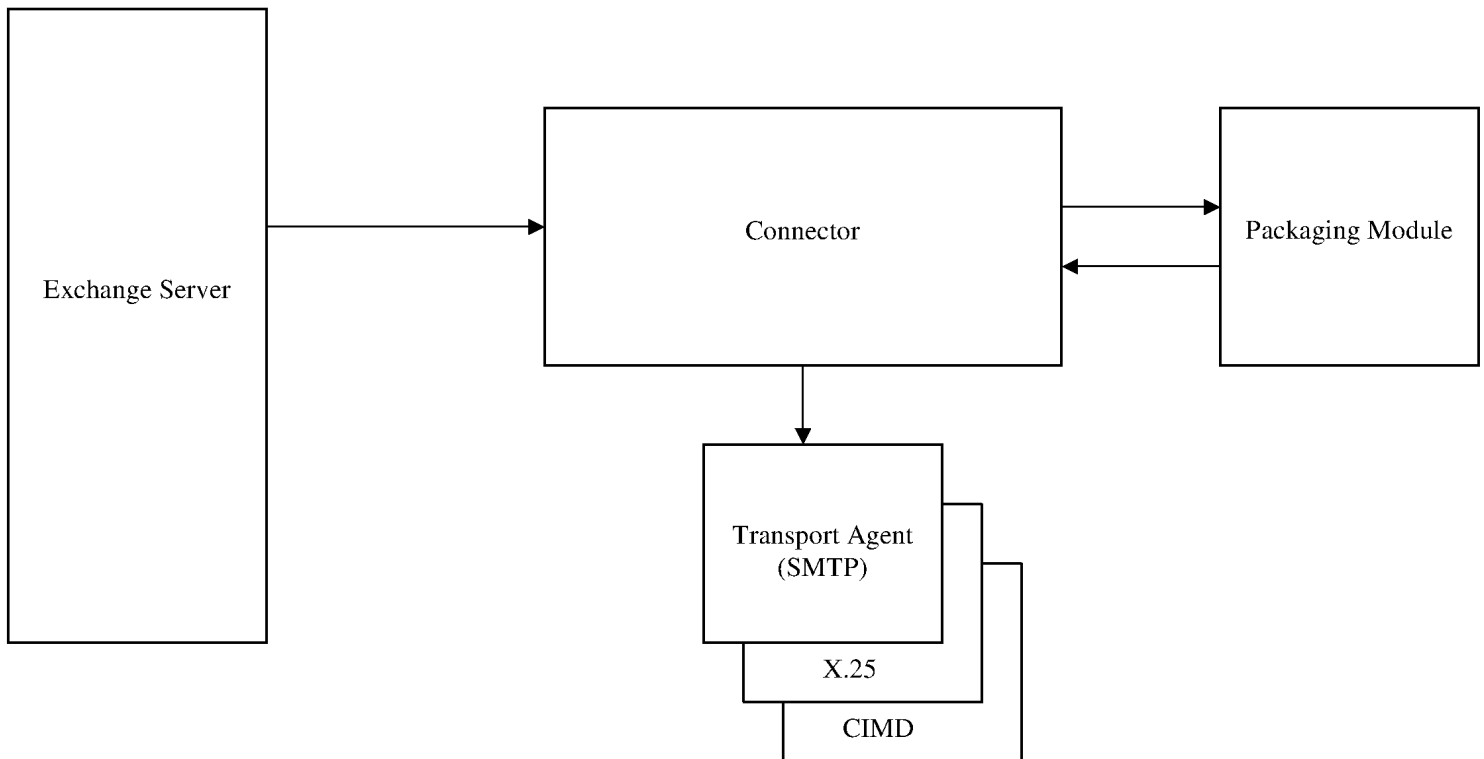
<

>

OK

Cancel

## Connector Architecture



### Connector

The connector communicates with the exchange server using HTTP/DAV and CDO (possibly other mechanisms?). To perform its task, it needs three pieces of data from the server:

- Information to be sent
- An address (email address, pin number, SMS address, etc.)
- The service provider

This information could be forwarded to the connector by a rule or some other mechanism could cause the connector to poll for this data. The information is used (processed) by the Packaging Module, while the address is used by the Transport Agent to get the information to the device. The service provider allows the connector to determine which Packaging Module and which Transport Module to use. If a service provider supports multiple transports, the address is used to determine which transport to use.

It is likely that Microsoft (our group) will implement the connector. It may not be necessary for the connector to be implemented as a COM object, however, the packaging module and transports will, most likely, be implemented as COM objects.

## Packaging Modules

These are the modules that take generic data from the Exchange Server, such as email, appointments, or other PIM data, and format it for a particular device/service. (Do we want to separate devices from services? This would mean having a Device Packaging Module and a Service Packaging Module as separate components. I.E. SMS and Nokia 9000 would be two modules that are called sequentially?) These modules would do things such as:

- Truncate a message to 160 chars for SMS (or break into multiple SMS messages)
- Formulate appointments into ICal or other Internet standards

It is likely that these modules will be written by the device manufacturers or by the service providers, however, we need to make the interfaces general enough for standard modules to be created. These modules will be implemented as COM objects to simplify the addition of new modules.

## Data Conversion

Some data items, such as file attachments being sent to Handheld PCs, may require conversion before being sent to the device (Do we even care about this scenario for this connector? Is sending attachments really a requirement for wireless? Probably yes, but something to be considered). HTTP/DAV should take care of this conversion by presenting the information as a different MIME body type, however this needs to be further investigated, and if it is not appropriately handled by DAV, the Packaging module needs to perform these conversions.

## Transport Agents

The transport agents take the data prepared by the packaging module and deliver it to the specified address.

These will be implemented as COM objects to simplify the interface.

Microsoft will implement the SMTP interface because it will be the most common interface and we can then use this as a sample for other companies to use in writing their transports.

## Data Flow

The connector will start when it receives a message, service provider, and address (as the result of a rule?). It then makes the necessary callbacks to the server to obtain any additional information it needs, such as information to compute a new synchronization token. This information is then passed to the appropriate packaging agent to be processed for the user's device and carrier (do we need to store the device type in Exchange and pass it with the rest of the information?). The packaging agent then passes the processed data back to the connector, which then passes it to the appropriate transport agent. The connector determines which transport agent to use based on the specified address and the carrier. (There also exists the possibility of letting the packaging module decide on and then communicate directly with the transport agent, however this may be too limiting)

## Configuration

Some of these modules, in particular, the transport agents, may need configuration information. Some of these, such as the SMTP or X.25 transport agents, will be using standard mail messages for getting data to the carrier. While these will probably use the Exchange Server they're installed on, there may be cases where a different mail server/port need to be specified (Do we want these objects talking directly to the server or force communication to come back through the connector?). Additionally, there may be other transports, such as an HTML based protocol (similar to connecting to a web page to send an alphanumeric page) where a proxy or html site need to be specified. As a first pass, we should require each of these objects to expose a configuration interface that would put up the appropriate UI for obtaining this information. The manner in which this interface is called, whether through server configuration, connector configuration, or a separate control panel remains an open issue.

## **EXHIBIT D**

General

Summary

Statistics

Contents

Custom

Created: Wednesday, May 20, 1998 9:20:02 AM

Modified: Friday, June 02, 2006 11:38:14 AM

Accessed: Tuesday, June 13, 2006 4:58:13 PM

Printed: Thursday, June 11, 1998 3:54:07 PM

Last saved by: Marc Seinfeld

Revision number: 53

Total editing time: 3430 Minutes

Statistics:

Statistic name	Value
Slides:	15
Paragraphs:	101
Words:	809
Bytes:	71449
Notes:	0
Hidden slides:	0
Multimedia clips:	0
Presentation format:	On-screen Show

OK

Cancel

# PT Wireless

Marcse/susannel

6/15/2006

Microsoft Confidential

# The Wireless Group

- ▶ Components covered
  - ▶ SMS, CE sync to server, RVP& NT SMS integration for internal customers, WAP response
- ▶ Team contacts
  - SMS : Dev - nfishman + : Test - audreys/trevor/stella : PM - marcse
  - CE sync : Dev - CE georgehu : Test - CE Ken Kiesow : PM - Susanne/CE's stevefla



# High level goals - PT

- ▶ SMS connectivity
  - ▶▶ pragmatic - 'primes the wireless data pump' & gets carriers to true wireless data (circuit w/QuickNetConnect & packet)
  - ▶▶ sync aware (benefit for smart devices w/storage and sync)
  - ▶▶▶ 1 & 2 way
  - ▶▶ multiple modular network support (some gsm, some cdma, some pager, some instant msg.; COM object so others can be added)
- ▶ Great CE dav sync directly with PT server in the box
  - ▶▶ CE sync to existing EX/Mapi servers via PUMA,
- ▶ HTML to micro-browser for mail, cal, contacts, tasks

# SMS connectivity - Specifics

- ▶ M2 clarify staffing and division of labor by 7/15
  - ▶ MS does dev for base connector
  - ▶ Publishable COM interfaces either contracted or done internally
  - ▶ network/transport modules either contracted and/or co-developed with carriers
  - ▶ device modules shopped out to device manufactures (we'll do a default) based on Nokia LOI template for network module

# SMS connectivity - Dates

- ▶ Base messaging limping by 8/31 (inside M2)
  - ▶▶ 2 way gsm w/Nokia (over IP, w/default formatting)
  - ▶▶ COM interfaces partially impl. in base connectivity piece
- ▶ COM interface specs closed 8/15
- ▶ specs for M3 work done 8/15 including support for:
  - ▶▶ gracefully handling 2 mailbox problem
  - ▶▶ message chaining, content conversion, admin, setup, event logging
- ▶ M3 dev work
  - ▶▶ default formatting module
  - ▶▶ coding on 'decided set' of network/transport modules
  - ▶▶ coding on base connectivity piece which is spoken to via DAV over IP from Exchange and other servers (like CE's sync/doc conversion, sql, etc.)

# SMS connectivity - Issues

- ▶ European v. US prioritization of networks, testing
- ▶ technically each SMSC has a different API



# Internal Teams interested in SMS

- ▶ Exchange RT RVP
- ▶ NT alert paging

No dates yet. Including them in spec. discussions so we can insure we support RT and paging requirements they encounter. Possible to include these in the 'decided set' particularly if contracted, in parallel to base development

# CE side Sync Goals

- ▶▶ Server side sync efficiency
  - ▶▶ Object based, not session based
- ▶▶ Common sync interface to a *variety* of data stores (so we're converging on DAV)
- ▶▶ Minimize code on the device side
- ▶▶ Modular design
  - ▶▶ Minimizes application redundancies
  - ▶▶ Reduces memory footprint
- ▶▶ numerous others which lead to issues ('all things to all people investigate mode', and ship dates for next Uber sync release currently beyond PT - 8/30/99)

# Server side Sync Requirements

- ▶ Supports Mail, Cal, Contacts, Tasks +
- ▶ Scalable, reliable, secure
- ▶ Server side conversion
  - ▶▶ Must be extensible
  - ▶▶ Weaning... Transition to standard formats (HTML, XML, iCal, vCard)
- ▶ Server side filtering
  - ▶▶ remotable user control to suit connection method
  - ▶▶ Progressive sync - nice to have

# The two teams jointly driving PT requirements

- ▶ DAV to write a spec xx/xx/98 for DAV-XML representation of mail, calendar, contacts, tasks
- ▶ clarifying whether DAV maintains a manifest/mapping table/collection blob/replication token, etc. per folder or per hierarchy. (if per folder the client must do more)
- ▶ DAV to consider progressive replication in manifest (changes before deletes, so they can be prioritized or skipped over costly links)
- ▶ to support the "sync only last n days" scenario
- ▶ Blob 'Minus' support - facilitates the "sync only last n days" scenario



# Issues - currently many

- ▶ Requirements not fully defined (MRD7/13/98)
  - ▶▶ What servers, & which versions
  - ▶▶ Replication scenarios (device-device, device-desktop, in addition to device-server)
  - ▶▶ Administration (devices, profiles, applications, backup, CE ZAW services)
- ▶ Dependencies
  - ▶▶ DAV, schemas, device OS...
  - ▶▶ Driving requirements and getting support across groups
  - ▶▶ Schedule conflicts across/within groups
  - ▶▶ Getting client apps bought into design/plan (internal CE org issues)
- ▶ Nexus (next Uber sync) resource issues
  - ▶▶ Waiting on Dev buy-in
  - ▶▶ Dev team working on WECS (Minerva) 2.2 - RTM 8/98
- ▶ Schedule
  - ▶▶ Dependency on Cedar (CE OS 3.0) - dates not firm. We believe server sync could be a separable piece that could ship directly in PT box.

# Plan B - the parachute

- ▶ rip cord pulled 8/1 if need be
- ▶ spec for modular PT server sync only final 8/15

# WAP response - HTML/XML to microbrowsers

[Redacted]

# BackUP slides

6/15/2006

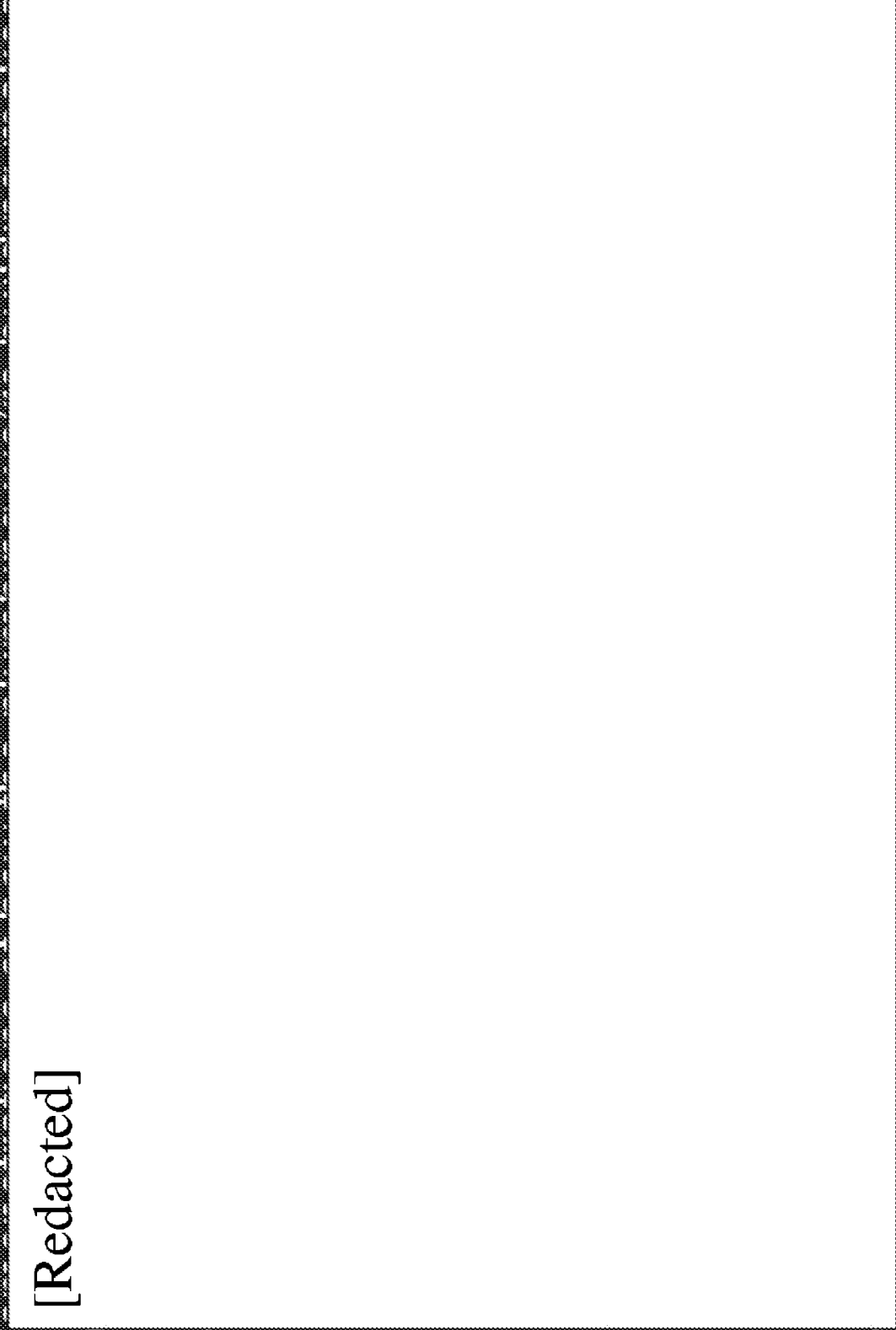
Microsoft Confidential

14

# Intern Projects - this summer

- ▶ d li a

peoples



n the

, or the

- ▶ d

- ▶ Dec 15/2018 and nation, 100% Microsoft Confidential consumer

## **EXHIBIT E**



General

Summary

Statistics

Contents

Custom

Created: Thursday, October 08, 1998 8:18:00 AM

Modified: Friday, June 02, 2006 11:38:13 AM

Accessed: Tuesday, June 13, 2006 4:49:38 PM

Printed: Friday, December 11, 1998 1:50:00 PM

Last saved by: Marc Seinfeld

Revision number: 1013

Total editing time: 5072 Minutes

Statistics:

Statistic name	Value
Pages:	31
Paragraphs:	1494
Lines:	2345
Words:	13669
Characters:	76351
Characters (with spaces):	89107
Bytes:	25600

---

# DMI Gateway Module

---

*Draft Status*

*Microsoft Confidential – Internal Use Only*

**Copyright © Microsoft Corporation, 1998. All Rights Reserved**

## 1. DMI Service Module

Spec Title	DMI Service Module
Component	Connectivity
Feature area	Wireless
Feature scope	Architectural
Related sections	<<related sections>>
Reference Sources	Wireless end-to-end.doc
Product Version	Platinum
Author	Don Kadyk
Feature Team	PM: marcse; Dev: leifp, nfishman, donk, ericsten, haoyu; Test: stellac
Spec status	Un-Reviewed
Spec stability	Unstable
Last Changed	11/12/98

Revision Summary		
Date	Author	Changes
12/10/98	DonK	Initial draft
13/11/98	DonK	Major updates
11/12/98	DonK	Final Draft

### 1.1 OVERVIEW

The DMI provides the basic HTTP input and output for the DMI. The DMI service module handles, among other things, IP connections for inbound and outbound messages, thread pooling, and performance counters.

Strictly speaking the DMI is an HTTP Proxy server for wireless devices. (See the definitions of these terms in section 1.4.)





<b>COPYRIGHT © MICROSOFT CORPORATION, 1998. ALL RIGHTS RESERVED.....</b>	<b>1</b>
<b>1. DMI SERVICE MODULE .....</b>	<b>1</b>
1.1 OVERVIEW.....	1
1.2 GOALS & OBJECTIVES .....	3
1.3 EXECUTIVE SUMMARY .....	3
1.4 NOTATIONAL CONVENTIONS .....	3
1.4.1 Rules .....	3
1.4.2 Basic Rules.....	3
1.5 DEFINITIONS AND TERMS .....	3
<b>2. DESIGN.....</b>	<b>3</b>
2.1 OVERALL DESIGN .....	3
2.2 NT SERVICE INTERFACE.....	3
2.3 EXCEPTION HANDLING.....	3
2.4 EVENT LOGGING.....	3
2.5 PERFORMANCE COUNTERS .....	3
2.6 THREAD POOL MANAGER.....	3
2.7 MESSAGE PROCESSOR .....	3
2.7.1 Message processor creation phase .....	3
2.7.2 Message processor initialization phase .....	3
2.7.3 Message processor locator phase .....	3
2.7.4 Message processor processing phase .....	3
2.7.5 Message processor deactivation phase.....	3
2.7.6 Message processor destruction phase.....	3
2.7.7 Message processing chains.....	3
2.7.8 External Interfaces and Structures .....	3
2.8 LOCALIZATION .....	3
2.9 REGISTRY ENTRIES.....	3
2.10 DEBUG FRAMEWORK.....	3
<b>3. DESIGN OPTIONS.....</b>	<b>3</b>
3.1 THREAD POOLING.....	3
3.2 INBOUND TCP CONNECTIONS .....	3
3.2.1 IIS.....	3
3.3 OUTBOUND UDP CONNECTIONS .....	3
3.4 INTERNAL QUEUES .....	3
<b>4. REFERENCES .....</b>	<b>3</b>
<b>5. OPEN ISSUES .....</b>	<b>3</b>

## 1.2 GOALS & OBJECTIVES

- Reliability – The DMI must be 100% reliable in message delivery to the wireless network.
- Scalability – The DMI is a function provider first. All due diligence **MUST** be made to provide a server that is scalable to a high number of users and messages per minute. However the first cut **MUST** provide all the functionality needed to insure the success of the product. Once that is complete performance monitoring and optimizations will be made to achieve the highest possible throughput.
- Provide a performance collection framework for the DMI
- Provide a Exception handling framework for the DMI
- Provide thread pooling for the DMI
- Provide routing control for device and generic content formatting chaining.

## 1.3 EXECUTIVE SUMMARY

<< Summary of design, features, architecture, etc>>

## 1.4 NOTATIONAL CONVENTIONS

This specification uses an augmented Backus-Naur Form (BNF) notation. The differences from standard BNF involve naming rules and indicating repetition and "local" alternatives.

### 1.4.1 Rules

#### 1.4.1.1 Rule Naming

Quotation marks enclose literal text (which may be upper and/or lower case). Certain basic rules are in uppercase, such as SPACE, TAB, CRLF, DIGIT, ALPHA, etc. Angle brackets are used in rule definitions, and in the rest of this document, whenever their presence will facilitate discerning the use of rule names.

#### 1.4.1.2 Rule1 / Rule2: Alternatives

Elements separated by slash ("/") are alternatives. Therefore "foo / bar" will accept foo or bar.

#### 1.4.1.3 (Rule1 Rule2): Local Alternatives

Elements enclosed in parentheses are treated as a single element. Thus, "(elem (foo / bar) elem)" allows the token sequences "elem foo elem" and "elem bar elem".

#### 1.4.1.4 \*RULE: REPETITION

The character "\*" preceding an element indicates repetition. The full form is:

<[>\*<m>element

indicating at least <[> and at most <m> occurrences of element. Default values are 0 and infinity so that "(element)" allows any number, including zero; "1\*element" requires at least one; and "1\*2element" allows one or two.

#### 1.4.1.5 [RULE]: OPTIONAL

Square brackets enclose optional elements; "[foo bar]" is equivalent to "1(foo bar)".

#### 1.4.1.6 Nrule: Specific Repetition

"<n>(element)" is equivalent to "<n>\*<n>(element)"; that is, exactly <n> occurrences of (element). Thus 2DIGIT is a 2-digit number, and 3ALPHA is a string of three alphabetic characters.

#### 1.4.1.7 #Rule: Lists

A construct "#" is defined, similar to "\*", as follows:

<l>#<m>element

indicating at least <l> and at most <m> elements, each separated by one or more commas (","). This makes the usual form of lists very easy; a rule such as '(element \*(", " element))' can be shown as "1#element". Wherever this construct is used, null elements are allowed, but do not contribute to the count of elements present. That is, "(element),,(element)" is permitted, but counts as only two elements. Therefore, where at least one element is required, at least one non-null element must be present.

Default values are 0 and infinity so that "#(element)" allows any number, including zero; "1#element" requires at least one; and "1#2element" allows one or two.

#### 1.4.1.8 ; COMMENTS

A semi-colon, set off some distance to the right of rule text, starts a comment that continues to the end of line. This is a simple way of including useful notes in parallel with the specifications.

### 1.4.2 Basic Rules

The following table outlines the basic rules used throughout the document.

Rule	Definition	Decimal range / Note
CHAR	<any ASCII character>	0 -127
ALPHA	<any ASCII alphabetic character>	65 - 90 97 -122
DIGIT	<any ASCII decimal digit>	48 - 57
HEX	<DIGIT \ "a" \ "b" \ "c" \ "d" \ "e" \ "f" \ \ "A" \ "B" \ "C" \ "D" \ "E" \ "F" >	48 – 57, 65-70, 97-102
CTL	<any ASCII control character and DEL>	0 - 31, 127
CR	<ASCII CR, carriage return>	13
LF	<ASCII LF, linefeed>	10
SPACE	<ASCII SP, space>	32
HTAB	<ASCII HT, horizontal-tab>	9
<">	<ASCII quote mark>	34
CRLF	CR LF	
LWSP	*(SPACE / HTAB)	The sequence can be replaced by a single space.

## 1.5 DEFINITIONS AND TERMS

**MSMQ** – Microsoft Message Queue

**MTS** – Microsoft Transaction Server

**Inbound** – For the purpose of this document this refers to any connection or message coming from the wireless network bound for a server on the IP network. This is a departure from the definition given in [1]. This is to help clarify the message flow through the DMI since the DMI will act as a proxy server in some cases and a gateway server in others.

**Outbound** – For the purpose of this document this refers to any connection or message coming from the IP network bound for the DMI or wireless network. This is a departure from the definition given in [1]. This is to help clarify the message flow through the DMI since the DMI will act as a proxy server in some cases and a gateway server in others.

**Proxy** – An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy **MUST** implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering. Except where either transparent or non-transparent behavior is explicitly stated, the HTTP proxy requirements apply to both types of proxies. [1]

**Gateway** – A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. [1]

## 2. Design

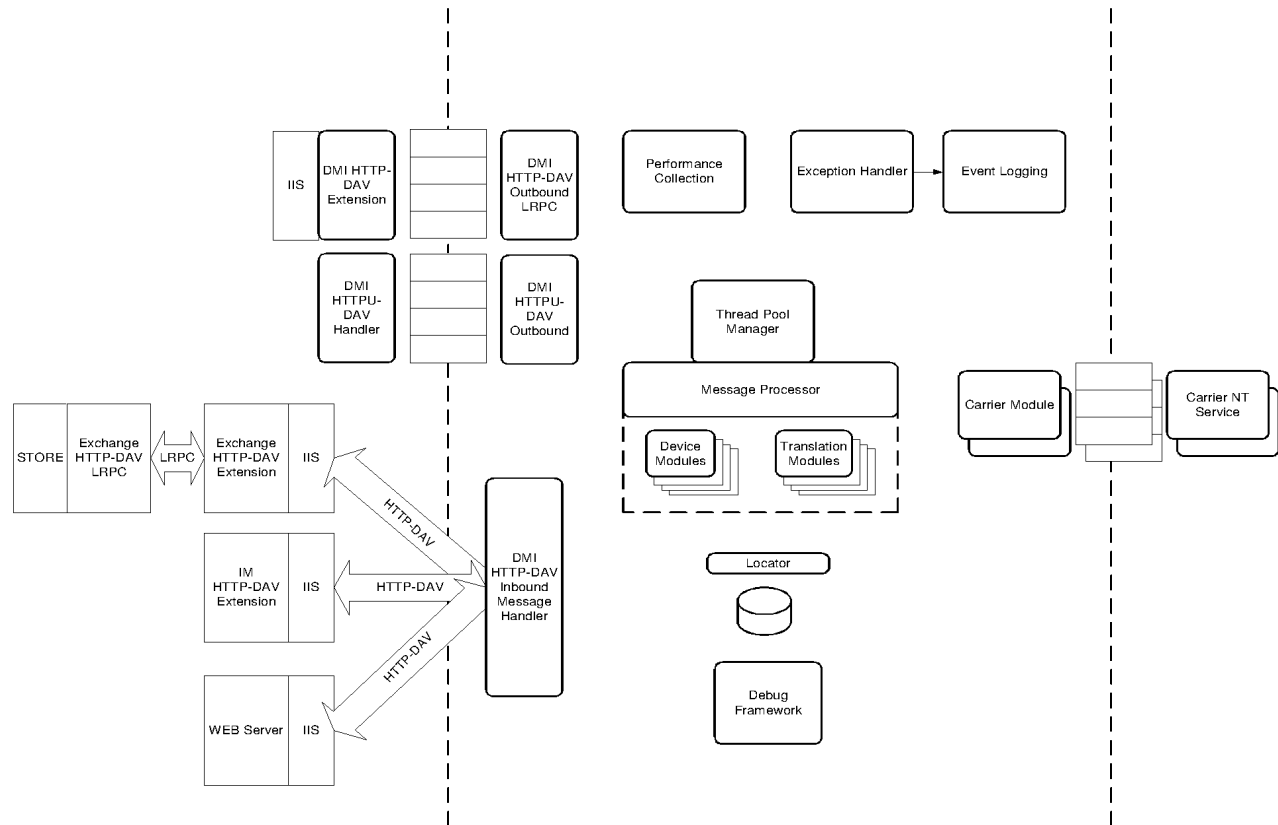
### 2.1 OVERALL DESIGN

The section covers the overall design of the DMI service. This service provides the basic framework for the DMI as a whole. It provides the threading, message processing control, address resolution, performance collection and message queuing.

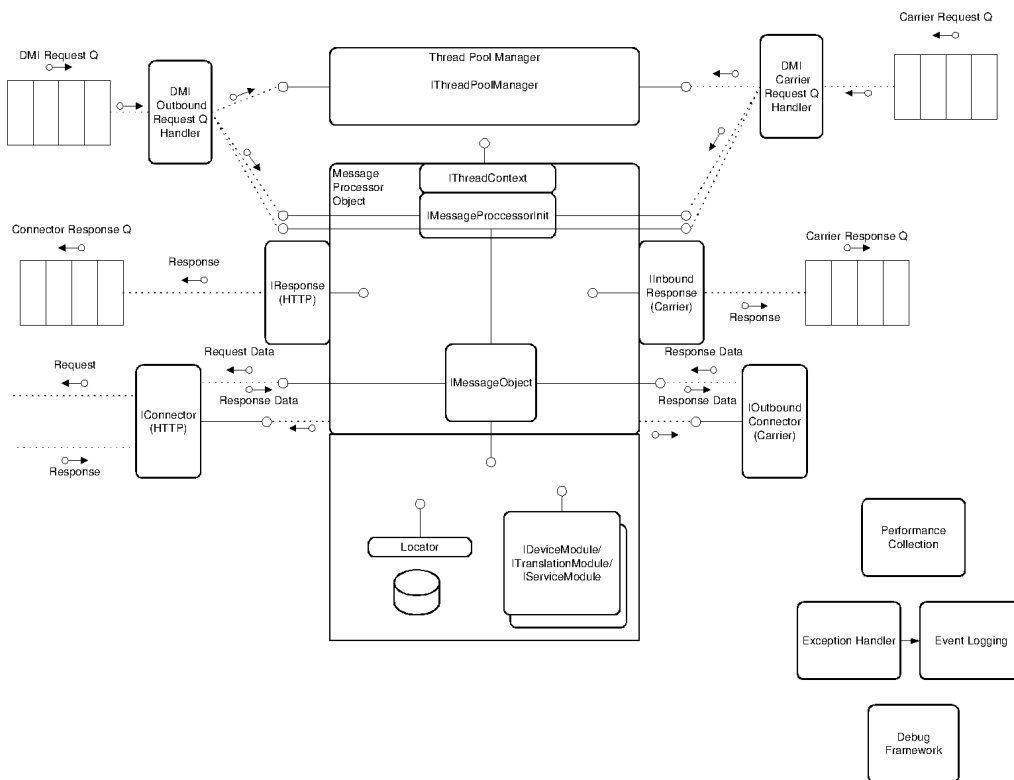
This program runs as an NT service. The service will be set-up as an automatic service so that it is available soon after the system has booted.

The following figure shows a big picture diagram of the DMI. This figure also shows some modules that are not part of the basic DMI to help in the understanding of the design.

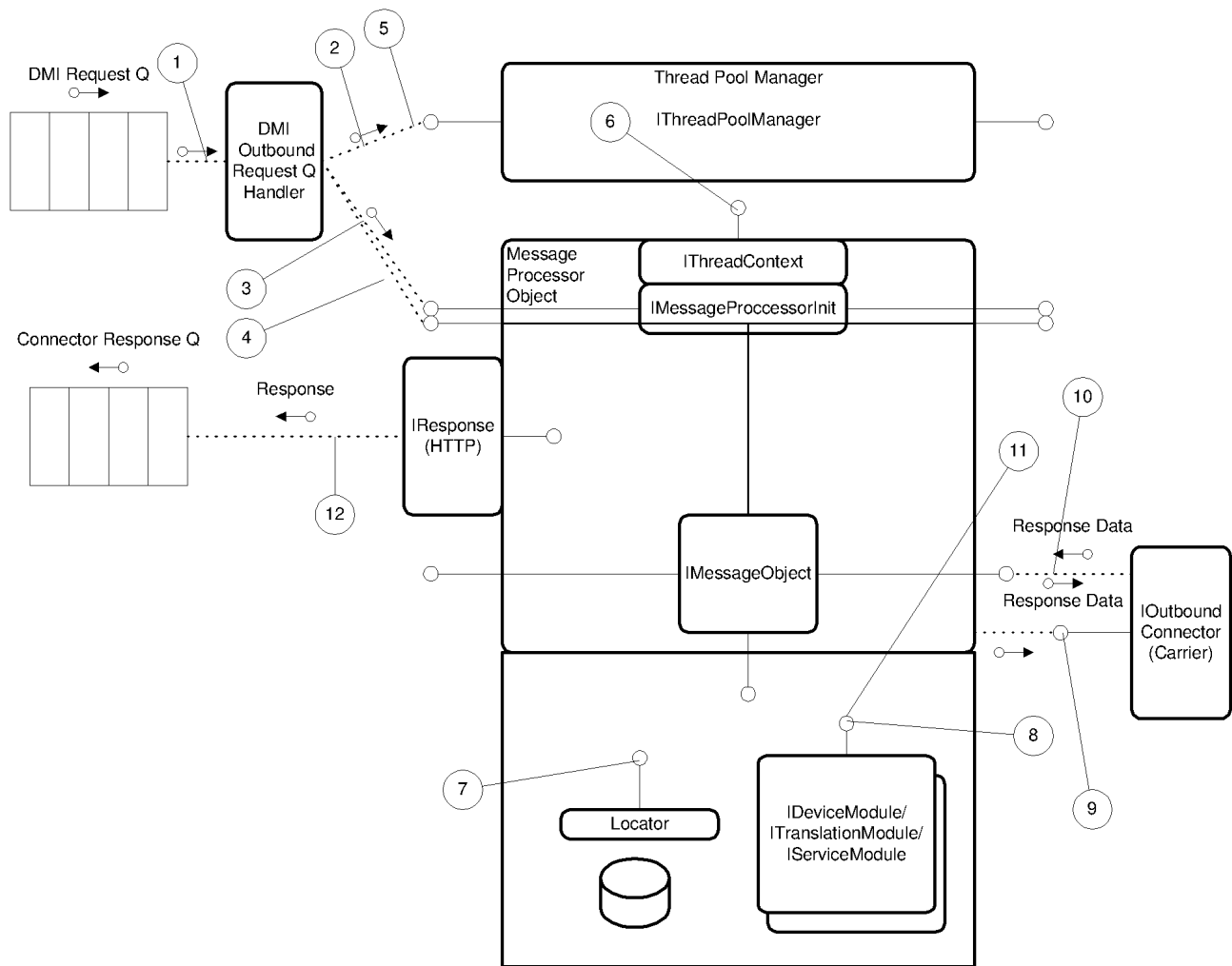
Employee



The following figure shows a more detailed view of the objects inside the DMI and the basic data flows in the DMI.



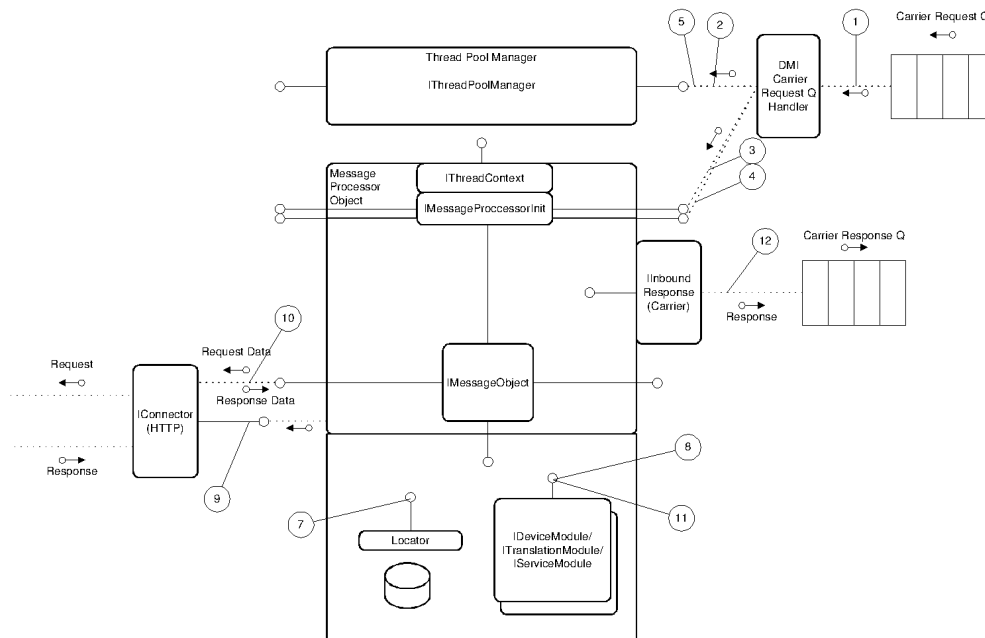
The following figure show the data flow through the DMI for a message traveling through to the in the outbound direction.



- 1) Message De-queued
- 2) Call thread pool manager to get handle to a Message Processor Object
- 3) Message Processor in initialized with response object
- 4) Message data in placed in the MessageObject.
- 5) Submit method called.
- 6) Free thread is found to run the Message processor object.
- 7) Message consults the Locator for needed information on how to process and route the message.
- 8) Message processor passes the MessageObject to the required modules.
- 9) Message processor hands the MessageObject to the outbound carrier module and blocks for return.
- 10) Carrier module read and sends the message and returns a response.
- 11) Message processor passes the MessageObject to the required modules if needed.
- 12) Response module is handed the MessageObject with and data is placed in the response queue.



The following figure show the data flow through the DMI for a message traveling through to the in the inbound direction.



- 1) Message De-queued
- 2) Call thread pool manager to get handle to a Message Processor Object
- 3) Message Processor in initialized with response object
- 4) Message data in placed in the MessageObject.
- 5) Submit method called.
- 6) Free thread is found to run the Message processor object.
- 7) Message consults the Locator for needed information on how to process and route the message.
- 8) Message processor passes the MessageObject to the required modules.
- 9) Message processor hands the MessageObject to the outbound carrier module and blocks for return.
- 10) DMI Inbound Connector read and sends the message and returns a response.
- 11) Message processor passes the MessageObject to the required modules if needed.
- 12) Response module is handed the MessageObject with and data is placed in the response queue.

## 2.2 NT SERVICE INTERFACE

The DMI service will act as an NT service. When running an NT service, such as this, a number of major issues need to be address.

The DMI will be setup to operate under a given user account instead of the “LocalSystem” account. The following table list the groups that the account must be a member of to operate properly.

Account Name	Type	Description
Server Operators	Security Group	Members can administer domain servers

Account Name	Type	Description
Schema Admins	Security Group – Global	Designated Administrators of the schema
Exchange Admins	Security Group – Global	Exchange Administrators

The following table lists the services that the DMI service is depends on and thus can not start until the listed services have started. Included are only those services that the DMI directly depends on and not the dependencies that the listed services have.

Service Name
Message Queuing Services (MSMQ)

The service handler function must respond to a number of control command issued by the Service Control Manager (SCM). These commands are issued by the SCM in response to various user or system actions. The following table shows how the service will respond to different commands from the SCM.

Command	Action VPt.0	Action VPt.1
SERVICE_CONTROL_STOP	Stops the DMI. All new connections are refused and pending connections are completed. The service will set the status to STOP_PENDING until all active connections have been completed and then change to STOPPED and exit.	Same
SERVICE_CONTROL_PAUSE	Not Handled	Service will pause. All pending messages will be completed. New connections will be refused. A PAUSE_PENDING status will be given until all current connections are completed. After that a PAUSED status will be given.
SERVICE_CONTROL_CONTINUE	Not handled	Service will resume from a pause. If a pause is currently pending it will be canceled.
SERVICE_CONTROL_INTERROGATE	Status Updated	Same
SERVICE_CONTROL_SHUTDOWN	Stops the DMI. All new connections are refused and pending connections are gracefully closed. The service will set the status to STOP_PENDING until all active connections have been closed and then change to STOPPED and exit. This process must take less than 20 seconds.	Same

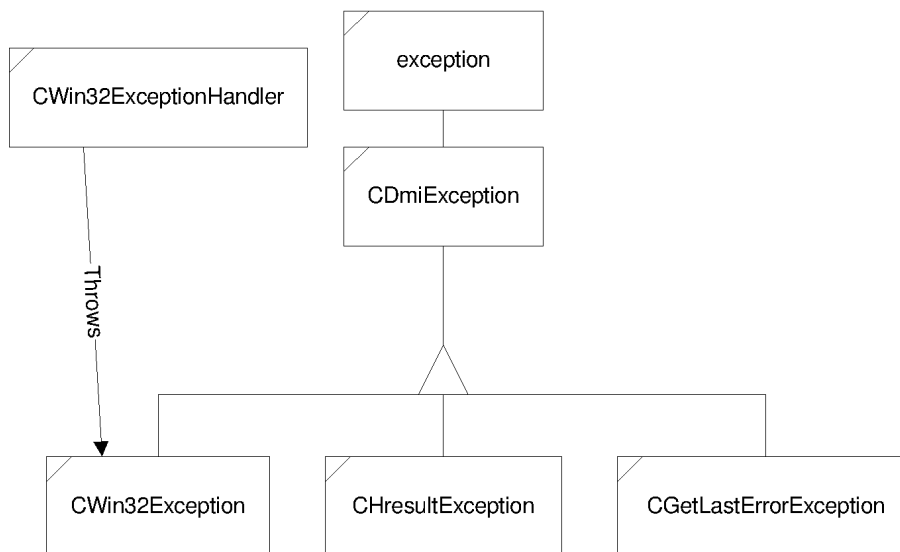
Command	Action VPt.0	Action VPt.1
SERVICE_CONTROL_PARAMCHANGE	Not handled	If the service is PAUSED this will re-read the service global parameters from the registry (or any INI/configuration files). This will cause the fresh values to take effect.
SERVICE_CONTROL_NETBINDADD	Not handled	Not handled
SERVICE_CONTROL_NETBINDREMOVE	Not handled	Not handled
SERVICE_CONTROL_NETBINDENABLE	Not handled	Not handled
SERVICE_CONTROL_NETBINDDISABLE	Not handled	Not handled

The service status will be updated whenever there is a change to the basic operating status of the service. In addition to this the status will be updated at the completion of all SCM commands, even if this does not change the status of the service.

## 2.3 EXCEPTION HANDLING

An exception-handling scheme must be designed that insures that all exceptions are handled properly. This should include Win32 exceptions (access violations, alignment faults, etc.) as well as those generated by the DMI.

To accomplish this the DMI will base all internal exception handling on the “exception” class defined in the “Standard C++ Library”. Given this as a foundation the following class hierarchy will be defined.



An instance of the Cwin32ExceptionHandler should be created at the top level of each thread of execution in the main DMI. This object will then catch all standard Win32 exceptions and translate and throw a Cwin32Exception class error.

A try/catch block is then used in all code that should catch and handle exceptions. At a minimum this should be the top-level point in all call trees. This top-level “catch” code should handle all exceptions and log in the event logs all errors that are not handled.

The CHresultException class provides a structured method for handling failures returned by COM objects (device, translation, and carrier objects).

The CGetLastErrorException class provides a structured method of handling Win32 and other failures that return error codes through the GetLastError().

## 2.4 EVENT LOGGING

The system will log events for all errors that cause exceptions, both system and DMI. In addition some form of enhanced logging will be required. Exactly what “enhanced” logging is will have to be further defined but the design should allow for at least the two levels of event logging.

The “verbose” logging would be turned off by default and on through some admin UI that would set a registry key. This registry key is defined in the section of this document that deals with registry settings.

## 2.5 PERFORMANCE COUNTERS

In all cases we shall be keeping the defined performance counters and values or we shall be able to derive them from ones already stored. The following set of tables defines the counters that the system will keep.

Since these counters are not persistent, per user data will not be kept in the standard memory based performance database but will instead be kept in a jet database. See the section on Quota Data Collection for more information on per user data collection.

There are also cases where the data stored in the performance counters should be kept for long term performance analysis. For version 1 we rely on the performance monitor and its functions to provide this ability. In the version 2 time frame we should revisit this option and determine if this is a feature that should be added and what data should be saved. This approach will give us time to determine what the key parameters are that should be monitored in a long-term manner.

The following table covers counters that are global to the DMI.

Value name	Units	Description
Total Input Rate	Messages / Min	This represents the message-input rate for a single DMI from all sources. (Inbound + Outbound)
Total Input count	Message	This represents the message-input count for a single DMI from all sources. (Inbound + Outbound) This is a running count and will roll over at $2^{32}$ .
Total Output rate	Messages / Min	This represents the message-output rate for a single DMI from all sources. (Inbound + Outbound)
Total Output count	Messages	This represents the message-input count for a single DMI from all sources. This is a running count and will roll over at $2^{32}$ . (Inbound + Outbound)
Maximum message size	Bytes	This represents the largest message handles by the DMI from any source.

Value name	Units	Description
Average message size	Bytes	This represents the average message size that the DMI handles from all sources

This table defines the counters kept for internal queues. The queue functional name must be appended to the value name given in this table to allow the administrator to determine which queue the counter is for.

Value name	Units	Description
Input rate	Messages/M in	This represents the message-input rate for a single DMI queue.
Input count	Messages	This represents the message-input count for a single DMI queue. This is a running count and will roll over at $2^{32}$ .
Output rate	Messages/M in	This represents the message-output rate for a single DMI queue.
Output Count	Messages	This represents the message-output count for a single DMI queue. This is a running count and will roll over at $2^{32}$ .
Max queue size	Messages	This represents the maximum number of messages in the queue.
Average queue Size	Messages	This represents the average number of message in the queue.
Max Time in queue	Seconds	This represents the maximum number of seconds that a message has waited in the queue.
Average Time in queue	Seconds	This represents the average number of seconds that a waits in the queue
Maximum message size	Bytes	This represents the largest message handles passed through the queue.
Average message size	Bytes	This represents the average message size passed through the queue.

The counters in the following table will be kept on any device modules or content converts in the system. The module identifier must be appended to the value name given in this table to allow the administrator to determine which module the counter is for.

For this purpose this includes any module that can be plugged into the message processor. This information is kept for each module and on a module basis. Thus, if a module performs many processing steps on a message the system will treat it as one step and provide only one set of performance data.

Value name	Units	Description
Messages processed count	messages	This represents total number of messages processed, both outbound and inbound. This is a running count and will roll over at $2^{32}$ .

Value name	Units	Description
Maximum message processing time	milliseconds	This represents the maximum time to process a message.
Average message processing time	milliseconds	This represents the average time to process a message.
Outbound messages processed count	messages	This represents total number of outbound messages processed. This is a running count and will roll over at $2^{32}$ .
Outbound message processing time	milliseconds	This represents the maximum time to process an outbound message.
Outbound average message processing time	milliseconds	This represents the average time to process an outbound message.
Outbound average size change ratio (output/input)	byte ratio	This represents the average message size change for outbound messages.
Outbound maximum size change ratio (output/input)	byte ratio	This represents the maximum message size change for outbound messages.
Outbound minimum size change ratio (output/input)	byte ratio	This represents the minimum message size change for outbound messages.
Inbound messages processed count	messages	This represents total number of inbound messages processed. This is a running count and will roll over at $2^{32}$ .
Inbound message processing time	milliseconds	This represents the maximum time to process an inbound message.
Inbound average message processing time	milliseconds	This represents the average time to process an inbound message.
Inbound average size change ratio (output/input)	byte ratio	This represents the average message size change for inbound messages.
Inbound maximum size change ratio (output/input)	byte ratio	This represents the maximum message size change for inbound messages.
Inbound minimum size change ratio (output/input)	byte ratio	This represents the minimum message size change for inbound messages.

The following table defines the counters that will be kept for each carrier module. The module identifier must be appended to the value name given in this table to allow the administrator to determine which module the counter is for.

Value name	Units	Description
------------	-------	-------------

Value name	Units	Description
Outbound messages processed count	messages	This represents total number of outbound messages processed. This is a running count and will roll over at $2^{32}$ .
Outbound maximum message processing time	milliseconds	This represents the maximum time to process an outbound message.
Outbound average message processing time	milliseconds	This represents the average time to process an outbound message.
Inbound messages processed count	messages	This represents total number of inbound messages processed. This is a running count and will roll over at $2^{32}$ .
Inbound message processing time	milliseconds	This represents the maximum time to process an inbound message.
Inbound average message processing time	milliseconds	This represents the average time to process an inbound message.

The following table of counters will be kept for each “Feature” This is an ill-defined category for now and we can better define it as we go.

*[Marcse: We need to know exactly what and how these measurements are to be made.]*

Value name	Units	Description
Message chaining usage count	messages	This represents total number of messages that have used the chaining feature. This is a running count and will roll over at $2^{32}$ .
Message chaining rate	messages / min	????
Inter Process Messages		I am not sure how to set this up but it would be nice to know the difference between those messages for user consumption and those messages for process consumption

## 2.6 THREAD POOL MANAGER

The thread pool manager handles the “Message Processor” objects as well as the threads that these objects do their work on.

The thread pool manager will be setup to handle an array of objects and threads. The size of these arrays will be configurable by means of system variables that may get their initial setting for the registry. The size of the array is determined at system start-up and is fixed thereafter. (A future version of the system will allow the system to be paused and the size of these arrays to be changed.)



The interfaces to the thread pool manager will not be exposed outside of the DMI. It will be accessed only from internal MS code. As such the exact details of this interface will be left to the header files to define. The header files that define this object are in TBD.

## 2.7 MESSAGE PROCESSOR

The message processor is a set of objects that operates on a message. The message processor is the component that is responsible for the address resolution as well as the message routing.

The message processor goes through a few stages of operation to process a message. The basic outline of those stages is outlined below with more details in the following sections.

Phase	Description
Creation	This happens during the DMI initialization phase. Once a message processor is created it is not destroyed until the DMI is shut down or some other event happens that would cause the message processor to be destroyed. It is during this phase that the message processor will allocate memory for handling messages. (While this memory allocation will happen initially based on some optimum message size. It must be possible for the memory to be grown. Once this memory is grown it will not be resized back to the optimum size.)
Initialization	In this phase the message processor is cleared of any remaining bits left behind by the last time it was used. Next the new message, response module, and flags are set for the next message to be processed.
Locator	In this phase the message data is analyzed to determine the message processor chain that will be used to process the message. Once the message chain is determined it is validated and loaded into the message processor. If any of the operations performed at this stage fail the message processor formulates a response and returns the response to the response module.
Process	In the process phase the message is processed through the chain. If a failure occurs at any point in the chain the message processor use the failure type and response message to return a failure response to the response module. Otherwise the message processor passes the message on to the <b>ICconnector</b> derived module that is associated with the chain. (See the section on Inbound and Outbound message processing for more details) Depending on the response from the <b>ICconnector</b> object the response if then either formulated and passed directly to the response module or it is processed back through the same chain in the reverse order and then handed to the response module.
Deactivation	In this phase the thread pool manager tells the message processor to release any releasable resources and prepare to be placed into an idle state.
Destruction	This phase is entered only when the system is being shut down or when some other event causes the message processor to be marked for destruction. It is at this time that the message processor releases any internal memory and ceases to exists.

Using this table as a guide the message processors lifetime can then be determined. The message processor begins its life in the “Creation” phase. At this point message objects are created and placed in a pool of objects maintained by the thread pool manager. While the DMI is running normally the message processor object go through a series of “Activation-Process-Deactivation” cycles. There are number of things that can cause the message processor objects to enter the “Destruction” phase. The first, and most common, in a DMI shutdown.

### 2.7.1 Message processor creation phase

The message processor objects are created and owned by the thread pool manager. The thread pool manager will create a list of message processor objects at DMI initialization time. The list of message processor objects will be used as outlined above.

The thread pool manager may chose to destroy a message processor object if the message processor if it detects a category III [2] failure in the message processor. This is the only other time that the thread pool manager will replace a message processor.

During this phase the message processor object will be initially “sized” by the thread pool manager. This initial “sizing” has to do with the **IMessageObject** that the message processor contains. The message processor will initially create an **IMessageObject** with the internal memory allocated to handle messages of a certain size or smaller with out allocating memory. The optimum size is set in the registry, see the section on registry entries for detailed information.

### 2.7.2 Message processor initialization phase

The message processor will expose the **IObjectControl** interface defined by the MTS system. This interface allows the thread manager to pool the message processor objects thus saving the cycles needed to create and destroy these objects. This will also allow the message processor to be run under MTS if this is required for resistance against **IMessageHandler** objects taking down the system. In the initial version this function shall be implemented without MTS.

The thread pool manager will call the message processors **IObjectControl::Activate()** just before calling any other methods that the message processor exposes. The thread pool manager will then call the **IObjectControl::Deactivate()** method after the message processor has completed its job. This will allow the message processor to prepare itself to process a new message as well as to any clean-up work after finishing processing a message.

In the **IObjectControl::Activate()** method the message processor clears the contained objects to insure that any previous message data is not accessible by any **IMessageHandler** objects.

After the thread pool manager has activated the message processor it will pass a pointer to the **IMessageProcessorInit** interface to the message queue handler that requested the thread. The message queue handler will call methods that set the message data, response handler, and control flags for the message to be processed.

When the message queue handlers are accessing the **IMessageObject** the contained objects are set to the following access levels. For more details on the design and working of the **IMessageObject** see the section covering the implementation details of the message processor implementation of that interface.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Orignal)	No Access
CMessageContainer   CMessageData (Request - Input)	Read / Write
CMessageContainer   CMessageData (Request - Output)	No Access
CMessageContainer   CMessageData (Response - Orignal)	No Access

CMessageObject contained object	Access
CMessageContainer   CMessageData (Response - Input)	No Access
CMessageContainer   CMessageData (Response - Output)	No Access
CInfoData (Message)	Read / Write
CInfoData (Device)	No Access
CInfoData (Carrier)	No Access
CInfoData (Preference)	No Access

### 2.7.3 Message processor locator phase

On a successful return from the message queue handler the thread pool manager will assign a thread for processing. The **IThreadContext** methods are called to start the thread processing the new message. Once the thread manager has started the message processor the thread pool manager is no longer involved in the message-processing task until the message processor completes its task.

The locator phase is, from the view of the message processor, started by allowing the locator to analyze the message and determines the message-processing chain of **IMessageHandler's** that must be called. How the locator does this is not a focus of this document. What is covered in this document is how the locator module passes information to the message processor and what the message processor does with this information.

The following is the information that the message processor passes to and expects from the locator. This in no way implies that the locator module should **only** supply this information, it may well supply information that the message-processor does not use but may be used by other modules in the chain. (See the user preferences section on the **CMessageObject** implementation for one example of other data that the locator provides.)

The following table shows the type of access that the Locator is given to the **CMessageObject**.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read
CMessageContainer   CMessageData (Request - Input)	Read
CMessageContainer   CMessageData (Request - Output)	Read / Write
CMessageContainer   CMessageData (Response - Original)	No Access
CMessageContainer   CMessageData (Response - Input)	No Access
CMessageContainer   CMessageData (Response - Output)	Write
CInfoData (Message)	Read / Write
CInfoData (Device)	Read / Write
CInfoData (Carrier)	Read / Write
CInfoData (Preference)	Read / Write

The following tables define the properties that must be set before a call to the locator. These properties are an expected minimum, this does not preclude more properties from being set. The first table represents the properties needed for an outbound message while the second is for inbound messages.

Property name and location for outbound messages	Value type	Description
<b>CMessageObject   CMessageContainer (Request)   CMessageData (Original)   CCPropertyData   ToUri</b>	BSTR	Holds the full URI of the devices HTTP address.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   FromIP</b>	BSTR	This is a standard dot formatted IP address of the server client sending the request. (Example: 192.168.234.142)

Property name and location for inbound messages	Value type	Description
<b>CMessageObject   CMessageContainer (Request)   CMessageData (Original)   CCPropertyData   Carrier</b>	BSTR	This holds the class instance name of the carrier class entry in the NTDS for the carrier that delivered the message request.
<b>CMessageObject   CMessageContainer (Request)   CMessageData (Original)   CCPropertyData   DeviceAddress</b>	BSTR	This holds the class instance name of the DeviceAddress class entry in the NTDS for the device that sent the message request.

The following tables are the message processor required information to be delivered by the locator. This is not to be considered all the information that the locator produces only the subset that is required by the message processor. For a complete definition of what the locator does see the section of this document that covers the locator implementation. There is a table for the inbound and outbound message case.

Property name and location for outbound messages	Value type	Description
<b>CMessageObject   CInfoData (Message)   CCPropertyData   ToUri</b>	BSTR	Holds the full URI of the devices HTTP address.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   User</b>	BSTR	This is the users NT account name.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   ProcessingChain</b>	BSTR	This is a complete copy of the message process chain. The format of this chain is given in the section in this document that covers these chains.

Property name and location for outbound messages	Value type	Description
<b>CMessageObject   CInfoData (Message)   CCPropertyData   DeviceHttpUri</b>	BSTR	Holds the full URI of the devices HTTP address. (This is the same thing as the ToUri.)
<b>CMessageObject   CInfoData (Message)   CCPropertyData   Carrier</b>	BSTR	This holds the class instance name of the carrier class entry in the NTDS for the carrier that delivered the message request.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   DeviceAddress</b>	BSTR	This holds the class instance name of the DeviceAddress class entry in the NTDS for the device that sent the message request.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   AccessRights</b>	DWORD	This is the devices access right. The format of this is defined is [3]
<b>CMessageObject   CInfoData (Message)   CCPropertyData   Connector</b>	BSTR	This is the CLSID of the IConnector object to use with the message.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   UserPreferanceLocator</b>	BSTR	This is a string that defines a method for retrieving the device / user settings and preferences. This value is used in an interaction between the CMessageObject and the CLocator. For further details see the implementation section for these objects.

Property name and location for inbound messages	Value type	Description
<b>CMessageObject   CInfoData (Message)   CCPropertyData   FromUri</b>	BSTR	Holds the full URI of the devices HTTP address.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   User</b>	BSTR	This is the users NT account name.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   ProcessingChain</b>	BSTR	This is a complete copy of the message process chain. The format of this chain is given in the section in this document that covers these chains.
<b>CMessageObject   CInfoData (Message)   CCPropertyData   DeviceHttpUri</b>	BSTR	Holds the full URI of the devices HTTP address. (This is the same thing as the FromUri.)

Property name and location for inbound messages	Value type	Description
<b>CMessageObject</b>   <b>CInfoData (Message)</b>   <b>CCPropertyData</b>   <b>Carrier</b>	BSTR	This holds the class instance name of the carrier class entry in the NTDS for the carrier that delivered the message request.
<b>CMessageObject</b>   <b>CInfoData (Message)</b>   <b>CCPropertyData</b>   <b>DeviceAddress</b>	BSTR	This holds the class instance name of the DeviceAddress class entry in the NTDS for the device that sent the message request.
<b>CMessageObject</b>   <b>CInfoData (Message)</b>   <b>CCPropertyData</b>   <b>AccessRights</b>	DWORD	This is the devices access right. The format of this is defined in [3]
<b>CMessageObject</b>   <b>CInfoData (Message)</b>   <b>CCPropertyData</b>   <b>Connector</b>	BSTR	This is the CLSID of the IConnector object to use with the message.
<b>CMessageObject</b>   <b>CInfoData (Message)</b>   <b>CCPropertyData</b>   <b>UserPreferenceLocator</b>	BSTR	This is a string that defines a method for retrieving the device / user settings and preferences. This value is used in an interaction between the CMessageObject and the CLocator. For further details see the implementation section for these objects.

In addition to locating the data outlined in the preceding tables it is expected that the message processing chain is complete. This means that the message processor will not attempt to chain in new modules based on any rules, the locator should take care of any extra processing rules. This requirement allows the message processor to be coded and remain fairly resilient to changes in chaining rules. The one thing that the message processor will handle in regard to message processing chains is to ensure that the chain is valid as outlined in [2].

#### 2.7.4 Message processor processing phase

Once the message processor has validated the chain as outlined in [2], it will determine if the message is a test message. If it is a test message it will perform the tasks outlined in [2].

The processing chains are given in the same order no matter which direction the message is going. The message processor calls the **IMessageHandler::OnMessage** methods in the chain. The chain order is always given in the outbound order. Thus the first module to get called and the order in which the modules will be called is determined by whether the message is in the request or response phase as well as the direction of the request message.

The following table is used to determine the first module and order the modules will be called. In this table the “HEAD” is the module toward the beginning of the chain and the “TAIL” is the end of the chain as seen in the string representation of a chain. The directions are given as “FORWARD” for processing the chain from “HEAD” to “TAIL” while “REVERSE” is for processing the chain from “TAIL” to “HEAD”.

Request/Response	Outbound/Inbound	First module	Direction
Request	Outbound	HEAD	FORWARD



Request/Response	Outbound/Inbound	First module	Direction
Response	Outbound	TAIL	REVERSE
Request	Inbound	TAIL	REVERSE
Response	Inbound	HEAD	FORWARD

The following table shows the type of access that the modules are given to the **CMessageObject** when called during a request phase.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read
CMessageContainer   CMessageData (Request - Input)	Read
CMessageContainer   CMessageData (Request - Output)	Read / Write
CMessageContainer   CMessageData (Response - Original)	No Access
CMessageContainer   CMessageData (Response - Input)	No Access
CMessageContainer   CMessageData (Response - Output)	Write
CInfoData (Message)	Read
CInfoData (Device)	Read
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

The following table shows the type of access that the modules are given to the **CMessageObject** when called during a response phase.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read
CMessageContainer   CMessageData (Request - Input)	Read
CMessageContainer   CMessageData (Request - Output)	Read
CMessageContainer   CMessageData (Response - Original)	Read
CMessageContainer   CMessageData (Response - Input)	Read
CMessageContainer   CMessageData (Response - Output)	Read / Write
CInfoData (Message)	Read
CInfoData (Device)	Read
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

As stated, the message processor calls the **IMessageHandler::OnMessage** for all modules in the chain. There are cases where the message processor will call other methods on derived interfaces of the **IMessageHandler**. The following section cover these interface their methods and when the message processor will call them.

#### 2.7.4.1 Standard error handling

The message processor performs the following algorithm in response to an error return from any called module in the message-processing phase. Returned errors are signaled by the

- 1) The message processor checks for a Status-Code property in the **CMessageObject** | **CMessageContainer** | **CMessageData (Response - Output)**. If this property is not set the message processor will set it based on the HRESULT returned from the module. See the table below for a mapping of HRESULT to Status-Codes.
- 2) The message processor then forms a full HTTP failure response in the **CMessageObject** | **CMessageContainer** | **CMessageData (Response - Output)**
- 3) If the message processor encountered the error during a outbound message the message processor calls the **IResponse::OnMessage** for the given message with the flags set to indicate a response.
- 4) If the message processor encountered the error during an inbound message the message processor run in the response-inbound mode and begins processing in the FORWARD direction from the **IDeviceModule**.

If the message processor encounters an error while in the standard error handling mode the event will be logged in the application log and the message processor will quit trying to handle the message.

The following table shows the standard mapping from HRESULT to Status code that the message processor imposes when a module returns a given HRESULT and does not set the Status-Code.

HRESULT	Status-Code	Description
EFAIL	500	Internal Server Error
(All error code not given in the table)	500	Internal Server Error
DMI_E_LOGINFAIL	403	Forbidden
DMI_E_LOGINREQUEST	401	Unauthorized
DMI_E_ACCESSDENIED	405	Method Not Allowed

#### 2.7.4.2 ILocator

**ILocator::GetPreferences** method is called whenever any module requests the user preference information. This request happens when a module calls the **IMessageObject::get\_ipPreferenceInfo** method and the preference information has not been yet retrieved. This information is retrieved the first time and cached locally for all requests after that on a per message basis.

The following table shows the type of access that the **ILocator::GetPreferences** method is given to the **CMessageObject** when called.

CMessageObject contained object	Access
<b>CMessageContainer</b>   <b>CMessageData (Request - Original)</b>	Read
<b>CMessageContainer</b>   <b>CMessageData (Request - Input)</b>	Read
<b>CMessageContainer</b>   <b>CMessageData (Request - Output)</b>	Read
<b>CMessageContainer</b>   <b>CMessageData (Response - Original)</b>	Read
<b>CMessageContainer</b>   <b>CMessageData (Response - Input)</b>	Read
<b>CMessageContainer</b>   <b>CMessageData (Response - Output)</b>	Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read

CMessageObject contained object	Access
CInfoData (Carrier)	Read
CInfoData (Preference)	Read / Write

The **CMessageObject** will keep an internal flag set to indicate that this event has occurred.

#### 2.7.4.3 ISecurityModule

The security modules have a special relationship with the message processor. The security module is allowed to perform a number of operations that other modules are not. These extra operations include the ability to specify a user to impersonate, change the message-processing chain, check the access rights of a device, and force user authentication. This extra functionality is afforded through the callbacks to the extra methods exposed by the **ISecurityModule**.

The message processor calls the **ISecurityModule::OnMessage** method as normal. When the method returns the return code will cause the message processor to continue processing as normal or take some other action based on the HRESULT returned. The following table outlines the actions taken by the message processor based on the return values from the **ISecurityModule::OnMessage** method.

HRESULT Returned from the OnMessage method	Action
(All error code not given in the table)	The message processor follows the standard error handling routine outline in a previous section.
DMI_OK_GETTOKEN	Message processor calls the <b>ISecurityModule::GetLogonToken</b> method
DMI_OK_GETCHAIN	Message processor calls the <b>ISecurityModule::GetChain</b> method
DMI_OK_CHECKACCESS	Message processor calls the <b>ISecurityModule::CheckAccessRights</b> method

Each of the extra methods called by the message processor causes the message processor to perform some extra function on return. Each of these functions is covered in the following sections.

##### 2.7.4.3.1 ISecurityModule::GetLogonToken

A successful return from this module causes the message processor to use the returned token in a call to the **ImpersonateLoggedOnUser** Win32 API function. This will cause the current message thread to run under the access right of the user. Before the message processor returns to the thread pool manager the message processor will call the **RevertToSelf** Win32 API function to return the thread to its own security context.

##### 2.7.4.3.2 ISecurityModule::GetChain

A successful return from this method causes the message processor to validate the chain passed back. If this chain is valid the message processor will start using the new chain beginning at the module indicated. If the chain is not valid the message processor will return a 500 Internal Server Error as outlined in the standard error handling using the original chain.

##### 2.7.4.3.3 ISecurityModule::CheckAccessRights

In this case the message processor will continue processing the message as normal if this method indicates a successful return. If the method indicates a failure then the message processor proceeds as outlined in the standard error handling section.

#### 2.7.4.4 IDeviceModule

**IDeviceModule::GetDeviceInfo** method is called whenever any module requests the device information. This request happens when a module calls the **IMessageObject::get\_ipDeviceInfo** method and the device

information has not been yet retrieved. This information is retrieved the first time and cached locally for all requests after that on a per message basis.

### 2.7.5 Message processor deactivation phase

After the message processor has returned from the thread pool manager will call the **IObjectControl::Deactivate()** method. In this function the message processor will clear the **CMessageObject** of all the data currently stored. In addition the message processor will release all interface pointers that it currently holds except for the **ILocator** and **IMessageObject** pointers.

### 2.7.6 Message processor destruction phase

As noted this phase only happens during shut down and is essentially the C++ de-constructor operation. In this phase the message object shall free all memory and the **CMessageObject** and the **CLocator** that it holds.

### 2.7.7 Message processing chains

The internal representation of the message processing chain is a double linked list. The data structure that is held in the list has the CLSID, flags, and pointer to the interface for each member in the chain. This structure is not available to the called modules and is maintained by the message processor.

The message processing chain is represented by a string in all representations outside of the message processor. The format of this string is outlined below.

message-processing-chain = "MPC"<version>"<chain flags>"1#<module entry>

version = <major version>"."<minor version> ; This is the version number of the  
; format no the chain

module entry = <CLSID in registry format><module-flags>

major version = 1\*<DIGIT>

minor version = 1\*<DIGIT>

chain-flags = <ascii encoded flags DWORD>

module-flags = <ascii encoded flags DWORD>

ascii encoded flags DWORD = "0x" 8\*<HEX>

CLSID in registry format = "{" String UUID "}"

String UUID = 8HEX "-" 4HEX "-" 4HEX "-" 4HEX "-" 12HEX ; See MSDN for more information

The <module entry> items in the chain are always given outbound order.

An example of a chain is given below. This is a version 1.0 chain with two modules.

**MPC1.0,0x3,{C9A6D8E6-912E-11d2-A750-00C04F79543B}0x1,{C9A6D8E6-912E-11d2-A750-00C04F79543B}0x2**

#### 2.7.7.1 Chain flags

The table below defines the meanings of the flags in the chain flags DWORD.

Bit	Description
<b>0</b>	Read
<b>1</b>	Read
<b>All other bits not listed</b>	Reserved

### 2.7.7.2 Module flags

The table below defines the meanings of the flags in the module flags DWORD.

Bit	Description
0	Module Type (See module type table for definition)
1	
2	
3	
All other bits not listed	Reserved

The following table defines the module types

Entry	Description
0	<b>ITranslatorModule</b> interface exposed
1	<b>IDeviceModule</b> interface exposed
2	<b>ISecurityModule</b> interface exposed
3	<b>IServiceModule</b> interface exposed
4	<b>ILocator</b> interface exposed (not supported in V1)
5	<b>IResponse</b> interface exposed (not supported in V1)
6	<b>IConnector</b> interface exposed (not supported in V1)
All other entries not listed	Reserved

## 2.7.8 External Interfaces and Structures

These interfaces and structures are defined in the MSDMI.IDL file.

Since some of these interfaces are public interfaces and it is expected that 3rd parties will write these modules we need some way to validate these objects both at run time and installation time. This method requirement is covered in detail in reference [2]

### 2.7.8.1 IMessageObject

This interface provides the methods used to pass the message data through the system. The message processor object creates and owns the **IMessageObject** object and an interface pointer to this object is passed to the **IMessageHandler::OnMessage()**.

The message processor create this object and passes it to device modules, translator modules, or carrier modules so that the module that it is passed to can operate on this message.

The object is implemented by containing objects based on one of two interfaces. The first is the **IMessageContainer** interface, and the second is the **IInfoData** interface. The **IMessageContainer**

interface provides access to various copies of the message data while the **IInfoData** interface provides access to various information about user preferences, device, carrier, or message.

#### When to implement

This interface is implemented by the DMI.

#### When to use

This interface is used to access the various information and message data for the current message being processed by the message processor. There are a number of objects that implement the **IMessageHandler** interface and it is through this interface that the message processor passes a pointer to this object.

#### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageObject::get_ipRequest</b>	Returns pointers to a contained object based on <b>IMessageContainer</b> that represents the request message that the DMI is currently processing.
<b>IMessageObject::get_ipResponse</b>	Returns pointers to a contained object based on <b>IMessageContainer</b> that represents the response message that the DMI is currently processing.
<b>IMessageObject::get_ipDeviceInfo</b>	Returns pointers to a contained object based on <b>IInfoData</b> that represents the device information for the device that the current message is from or going to.
<b>IMessageObject::get_ipMessageInfo</b>	Returns pointers to a contained object based on <b>IInfoData</b> that represents the message information that the DMI is currently processing.
<b>IMessageObject::get_ipCarrierInfo</b>	Returns pointers to a contained object based on <b>IInfoData</b> that represents the carrier/network information for the wireless network that the current device is bound to.
<b>IMessageObject::get_ipPreferenceInfo</b>	Returns pointers to a contained object based on <b>IInfoData</b> that represents the user preference information

#### 2.7.8.2 IMessageContainer

This interface provides methods used access three different copies of the message data. All three copies of the message data are presented through contained objects based on the **IMessageData** interface.

#### When to implement

This interface is implemented by the DMI.

#### When to use

This interface is used to access the various versions of the message data. You can get a pointer to one of these objects through the **IMessageObject**.

#### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.

Interface::Methods	Description
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageContainer::get_ipOriginal</b>	Returns pointers to a contained object based on <b>IMessageData</b> the original message placed into the input object. This copy of the message is set to read only access.
<b>IMessageContainer::get_ipInput</b>	Returns pointers to a contained object based on <b>IMessageData</b> . This object represents the input message to an <b>IMessageHandler</b> module. This copy of the message is set to read only access.
<b>IMessageContainer::get_ipOutput</b>	Returns pointers to a contained object based on <b>IMessageData</b> . This object represents the output message from an <b>IMessageHandler</b> module. This copy of the message is set for read/write access.
<b>IMessageContainer::CopyInToOut</b>	This method performs a quick copy of the input message data to the output message data.

### 2.7.8.3 IMessageData

This interface provides methods used access a single copy of the message data. The message is presented in the three parts. The first part is a **BASE\_FORMAT\_TYPE** structure that define the basic format used to represent in the next two sections of the message. The second part is the header information presented as a collection of properties. The final part is the message body and is presented as a stream.

#### When to implement

This interface is implemented by the DMI.

#### When to use

This interface is used to access a version of the message data. You can get a pointer to one of these objects through the **IMessageContainer**.

#### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageData::get_pFormatType</b>	Fills in a caller supplied <b>BASE_FORMAT_TYPE</b> structure.
<b>IMessageData::put_pFormatType</b>	Sets the internal <b>BASE_FORMAT_TYPE</b> structure.
<b>IMessageData::get_ipHeader</b>	Returns pointers to a contained object based on <b>IPropertyData</b> . This object holds header or property data associated with the message body or the protocol use to transport it. (HTTP headers are an example.)
<b>IMessageData::get_ipBody</b>	Returns a pointer to a contained object based on the <b>IStream</b> interface. This represents the message body.



#### 2.7.8.4 **IInfoData**

This interface provides methods used access a information data. The information is presented in the three parts. The first part is a **BASE\_INFO\_TYPE** structure that define the basic format used to represent in the next two sections of the message. The second part is the property information and is expected to hold the bulk of the data represented by this interface. The final part is the extended data and is presented as a stream. The extended data stream allows the system to handle information data, about the carrier for example, which does not fit into the property format.

##### **When to implement**

This interface in implemented by the DMI.

##### **When to use**

This interface is used to access a version of the message data. You can get a pointer to one of these objects through the **IMessageObject**.

##### **Methods in V-table Order**

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IInfoData::get_pInfoType</b>	Fills in a caller supplied <b>BASE_INFO_TYPE</b> structure.
<b>IInfoData::get_ipProperty</b>	Returns pointers to a contained object based on <b>IPropertyData</b> . This object holds information in the form of a property. What the properties are depends on the type of information stored.
<b>IInfoData::get_ipExtended</b>	Returns a pointer to a contained object based on the <b>IStream</b> interface. This represents the extended information that does not fit into the property model. MS will most likely not uses this in it base offerings.

#### 2.7.8.5 **IPropertyData**

This interface provides methods used access property based data. This interface inherits the **IPropertyBag** interface and adds the **IPropertyData::Delete()** and **IPropertyData::GetEnum()** to the methods provided. This allows for the full control of the properties contained in the object.

This interface is used to represent properties and header information in the DMI.

##### **When to implement**

This interface in implemented by the DMI.

##### **When to use**

This interface is used to access generic property data. You can get a pointer to one of these objects through the **IMessageData** or **IInfoData** interfaces.

##### **Methods in V-table Order**

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.

Interface::Methods	Description
<b>IPropertyBag::Read</b>	Called to read a property from the storage provided by the DMI.
<b>IPropertyBag::Write</b>	Called to write each property in turn to the storage provided by the DMI.
<b>IPropertyData::Delete</b>	Called to remove a property from the storage provided by the DMI.
<b>IPropertyData::GetEnum</b>	Called to get an enumeration interface ( <b>IEnumProperty</b> ) to the stored properties.

#### 2.7.8.6 IEnumProperty

This interface provides methods used enumerate through property based data stored in an **IPropertyData** object. This interface is one version of the **IEnumXXXX** interface defined by COM. The interface uses the **PROPERTY\_INFO\_TYPE** structure to pass property data to the caller.

##### When to implement

This interface is implemented by the DMI.

##### When to use

This interface is used to enumerate property data. You can get a pointer to one of these objects through the **IPropertyData** interfaces.

##### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IEnumProperty::Next</b>	Retrieves a specified number of items in the enumeration sequence.
<b>IEnumProperty::Skip</b>	Skips over a specified number of items in the enumeration sequence.
<b>IEnumProperty::Reset</b>	Resets the enumeration sequence to the beginning.
<b>IEnumProperty::Clone</b>	Creates another enumerator that contains the same enumeration state as the current one.

#### 2.7.8.7 IMessageHandler

This interface provides a set of methods that are called by the DMI's message processor. This interface is the main interface that is implemented by developers writing object to be used by the DMI. It contains the methods that must be implemented by all external objects.

The **IMessageHandler::OnMessage** does the major work of the object. It is called when the DMI message processor has determined that the object needs to act on the current message. The object is passed the message in the form of an **IMessageObject** interface as well as a set of flags that indicate what phase the message is in as well as the direction of the message.

##### When to implement

This interface should only be implemented when inherited into a different interface. (Handle it as though it were the **IUnknown** interface.)

#### When to use

The DMI message processor calls the methods of this interface.

#### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.

#### 2.7.8.8 IDeviceModule

This interface provides the interface that is needed to implement a device module. There can only be one of these modules per message. In addition to the methods of the **IMessageHandler** interface the device module includes a method **IDeviceModule::GetDeviceInfo** used by the DMI to get device information.

#### Message object control

The following table shows the type of access that is given to the **CMessageObject** when called during a request phase.

CMessageObject contained object	Access
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read / Write
<b>CMessageContainer   CMessageData (Response - Original)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Input)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Output)</b>	Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read / Write
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read
CMessageContainer   CMessageData (Request - Input)	Read
CMessageContainer   CMessageData (Request - Output)	Read
CMessageContainer   CMessageData (Response - Original)	Read
CMessageContainer   CMessageData (Response - Input)	Read
CMessageContainer   CMessageData (Response - Output)	Read / Write
CInfoData (Message)	Read
CInfoData (Device)	Read / Write
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

### Actions in message processing phases

These modules should perform device specific message translation or formatting in its implementation of the **OnMessage** method.

### When to implement

This interface should be implemented to provide device specific formatting for a message.

### When to use

The DMI message processor calls the methods of this interface.

### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.
<b>IDeviceModule::GetDeviceInfo</b>	Called by the DMI to retrieve the device information in an <b>IInfoData</b> object.

### 2.7.8.9 ITranslatorModule

This interface provides the interface that is needed to implement a translator module. There can be any number of these modules per message. There are no additional methods beyond that of the **IMessageHandler** interface. Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

### Message object control

The following table shows the type of access that is given to the **CMessageObject** when called during a request phase.

<b>CMessageObject contained object</b>	<b>Access</b>
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read / Write
<b>CMessageContainer   CMessageData (Response - Original)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Input)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Output)</b>	Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

<b>CMessageObject contained object</b>	<b>Access</b>
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read
<b>CMessageContainer   CMessageData (Response - Original)</b>	Read
<b>CMessageContainer   CMessageData (Response - Input)</b>	Read
<b>CMessageContainer   CMessageData (Response - Output)</b>	Read / Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

### **Actions in message processing phases**

Translator modules are to perform some sort of generic message translation in the **OnMessage** method. This may be to implement some form of generic compression or format translation.

### **When to implement**

This interface should be implemented to provide generic formatting or translation for a message.

### **When to use**

The DMI message processor calls the methods of this interface.

### **Methods in V-table Order**

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.

#### 2.7.8.10 IServiceModule

This interface provides the interface that is needed to implement a service module. There can be any number of these modules per message. There are no additional methods beyond that of the **IMessageHandler** interface. Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

##### Message object control

The following table shows the type of access that is given to the **CMessageObject** when called during a request phase.

CMessageObject contained object	Access
<b>CMessageContainer   CMessageData (Request - Orignal)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read
<b>CMessageContainer   CMessageData (Response - Orignal)</b>	Read
<b>CMessageContainer   CMessageData (Response - Input)</b>	Read
<b>CMessageContainer   CMessageData (Response - Output)</b>	Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

CMessageObject contained object	Access
<b>CMessageContainer   CMessageData (Request - Orignal)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read
<b>CMessageContainer   CMessageData (Response - Orignal)</b>	Read

CMessageObject contained object	Access
CMessageContainer   CMessageData (Response - Input)	Read
CMessageContainer   CMessageData (Response - Output)	Read
CInfoData (Message)	Read
CInfoData (Device)	Read
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

### Actions in message processing phases

Service modules are to perform some sort of service oriented functions in the **OnMessage** method. This may be to implement some form of quota logging and enforcing rules or a statistics gathering function.

### When to implement

This interface should be implemented to provide a generic service for a message.

### When to use

The DMI message processor calls the methods of this interface.

### Methods in V-table Order

Interface::Methods	Description
IUnknown::QueryInterface	Returns pointers to supported interfaces.
IUnknown::AddRef	Increments reference count.
IUnknown::Release	Decrements reference count.
IMessageHandler::OnMessage	Called when the message processor has a message to process.
IMessageHandler::AcceptFormats	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
IMessageHandler::ProduceFromats	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.

### 2.7.8.11 ISecurityModule

This interface provides the interface that is needed to implement a security module. There can only one of these modules per message. There are no additional methods beyond that of the IMessageHandler interface. Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

The message processor will handle returns from this interface in a different manor as set out in the section covering this module in the message processor message-processing phase.

### Message object control

The following table shows the type of access that is given to the **CMessageObject** when called during a request phase.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read



<b>CMessageObject contained object</b>	<b>Access</b>
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read / Write
<b>CMessageContainer   CMessageData (Response - Original)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Input)</b>	No Access
<b>CMessageContainer   CMessageData (Response - Output)</b>	Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

<b>CMessageObject contained object</b>	<b>Access</b>
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read
<b>CMessageContainer   CMessageData (Response - Original)</b>	Read
<b>CMessageContainer   CMessageData (Response - Input)</b>	Read
<b>CMessageContainer   CMessageData (Response - Output)</b>	Read / Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read
<b>CInfoData (Preference)</b>	Read

### **Actions in message processing phases**

The security module is able to effect greater control over message processing than other modules. This extra control is described in the message processing section of this document. It is expected that the security modules provide for authentication and encryption functions.

### **When to implement**

This interface should be implemented to provide a security for a message.

### **When to use**

The DMI message processor calls the methods of this interface.

### **Methods in V-table Order**

<b>Interface::Methods</b>	<b>Description</b>
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.

Interface::Methods	Description
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.
<b>ISecurityModule::GetLogonToken</b>	Called when the message processor detects that the <b>OnMessage</b> method has returned DMI_OK_GETTOKEN. Returns a token to the logged on user of the device.
<b>ISecurityModule::GetChain</b>	Called when the message processor detects that the <b>OnMessage</b> method has returned DMI_OK_GETCHAIN. Returns a message-processing chain that the message processor should use in further processing of the message.
<b>ISecurityModule::CheckAccessRights</b>	Called when the message processor detects that the <b>OnMessage</b> method has returned DMI_OK_CHECKACCESS. Returns an HRESULT indicating whether the message processor should continue to process the message or return a 405 method not allowed error to the user.
<b>ISecurityModule::TimerTick</b>	Called by a DMI service thread that allows the module to time out user login's.

#### 2.7.8.12 IResponse

This interface provides the interface that is needed to implement a response module. There can only be module derived from the **IResponse** interface per message. There are no additional methods beyond that of the **IMessageHandler** interface. This module is designed to be the in-process part of an outbound data source.

In V1 of the DMI product this is the module responsible for returning the response to the IIS Extension through its response queue.

Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

##### Message object control

The **IResponse** interfaces are not called during a request phase.

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

CMessageObject contained object	Access
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read

CMessageObject contained object	Access
CMessageContainer   CMessageData (Response - Original)	Read
CMessageContainer   CMessageData (Response - Input)	Read
CMessageContainer   CMessageData (Response - Output)	Read / Write
CInfoData (Message)	Read
CInfoData (Device)	Read
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

### Actions in message processing phases

This module is called only during the response phase of an outbound message. The module is responsible for sending the response to an outbound request back to the HTTP client that sent the message.

### When to implement

This interface should be implemented to provide the path back to the requestor of a message. This should be thought of in the HTTP 1.1 Request/Response sense although in some cases this will not be the wired/wireless protocol used to deliver the message.

### When to use

The DMI message processor calls the methods of this interface.

### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.

#### 2.7.8.13 IInboundResponse

This interface provides the interface that is needed to implement a response module. There can only be module derived from the **IResponse** interface per message. In addition to the methods of the **IMessageHandler** interface the device module includes a method **IInboundResponse::GetDeviceInfo** used by the DMI to get carrier information. This module is designed to be the in-process part of an inbound data source.

In V1 of the DMI product this is the module responsible for returning the response to the carrier through its response queue.

Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

## Message object control

The **IInboundResponse** interfaces are not called during a request phase.

The following table shows the type of access that is given to the **CMessageObject** when called during a response phase.

<b>CMessageObject</b> contained object	Access
<b>CMessageContainer</b>   <b>CMessageData</b> (Request - Original)	Read
<b>CMessageContainer</b>   <b>CMessageData</b> (Request - Input)	Read
<b>CMessageContainer</b>   <b>CMessageData</b> (Request - Output)	Read
<b>CMessageContainer</b>   <b>CMessageData</b> (Response - Original)	Read
<b>CMessageContainer</b>   <b>CMessageData</b> (Response - Input)	Read
<b>CMessageContainer</b>   <b>CMessageData</b> (Response - Output)	Read / Write
<b>CInfoData</b> (Message)	Read
<b>CInfoData</b> (Device)	Read
<b>CInfoData</b> (Carrier)	Read
<b>CInfoData</b> (Preference)	Read

## Actions in message processing phases

This module is called only during the response phase of an inbound message. The module is responsible for sending the response to an inbound request back to the device that sent the message. As noted this module is also required to fill in the carrier information when requested.

## When to implement

This interface should be implemented to provide the path back to the requestor of a message. This should be thought of in the HTTP 1.1 Request/Response sense although in some cases this will not be the wired/wireless protocol used to deliver the message.

## When to use

The DMI message processor calls the methods of this interface.

## Methods in V-table Order

<b>Interface::Methods</b>	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.
<b>IInboundResponse::GetCarrierInfo</b>	Called by the DMI to retrieve the carrier information in an <b>IInfoData</b> object.

#### 2.7.8.14 IConnector

This interface provides the interface that is needed to implement a connector module. There can only be one module derived from the **IConnector** interface per message. There are no additional methods beyond that of the **IMessageHandler** interface. This module is designed to handle inbound request/responses on the internal IP network.

In V1 of the DMI product this is the module responsible for the actual HTTP TCP/IP connections to the Intranet.

Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

##### Message object control

The **IConnector** interfaces are only called during an inbound request phase. However it is the last module called in this phase and may be viewed as the first module in the inbound response phase. However when it is called the message processor will indicate the phase is inbound request and will change to the inbound response phase on return from this module's **OnMessage** method.

CMessageObject contained object	Access
CMessageContainer   CMessageData (Request - Original)	Read
CMessageContainer   CMessageData (Request - Input)	Read
CMessageContainer   CMessageData (Request - Output)	Read
CMessageContainer   CMessageData (Response - Original)	Read
CMessageContainer   CMessageData (Response - Input)	Read
CMessageContainer   CMessageData (Response - Output)	Read / Write
CInfoData (Message)	Read
CInfoData (Device)	Read
CInfoData (Carrier)	Read
CInfoData (Preference)	Read

##### Actions in message processing phases

As noted this module is called at the end of an inbound request phase and in the **OnMessage** method it is expected that this module will produce or acquire the response to the message.

##### When to implement

This interface should be implemented to provide the intranet/internet connection for the request/response of the message set by a device. This should be thought of in the HTTP 1.1 Request/Response sense although in some cases this will not be the wired/wireless protocol used to deliver the message.

##### When to use

The DMI message processor calls the methods of this interface.

##### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.

Interface::Methods	Description
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.

#### 2.7.8.15 IOutboundConnector

This interface provides the interface that is needed to implement a connector module. There can only be module derived from the **IConnector** interface per message. In addition to the methods of the **IMessageHandler** interface the device module includes a method **IOutboundConnector::GetDeviceInfo** used by the DMI to get carrier information. This module is designed to handle outbound request/responses on the carrier network that handles the device.

In V1 of the DMI product this is the module responsible for the actual HTTP TCP/IP connections to the Intranet.

Implementing this interface implies certain responsibilities and restrictions regarding what the module is to do when called on to process a message.

##### Message object control

The **IConnector** interfaces are only called during an outbound request phase. However it is the last module called in this phase and may be view as the first module in the outbound response phase. However when it is called the message processor will indicate the phase is outbound request and will change to the outbound response phase on return from this module's **OnMessage** method.

CMessageObject contained object	Access
<b>CMessageContainer   CMessageData (Request - Original)</b>	Read
<b>CMessageContainer   CMessageData (Request - Input)</b>	Read
<b>CMessageContainer   CMessageData (Request - Output)</b>	Read
<b>CMessageContainer   CMessageData (Response - Original)</b>	Read
<b>CMessageContainer   CMessageData (Response - Input)</b>	Read
<b>CMessageContainer   CMessageData (Response - Output)</b>	Read / Write
<b>CInfoData (Message)</b>	Read
<b>CInfoData (Device)</b>	Read
<b>CInfoData (Carrier)</b>	Read / Write
<b>CInfoData (Preference)</b>	Read

##### Actions in message processing phases

As noted this module is called at the end of an outbound request phase and in the **OnMessage** method it is expected that this module will produce or acquire the response to the message. As noted this module is also required to fill in the carrier information when requested.

##### When to implement

This interface should be implemented to provide the carrier connection for the request/response of the message set by a device. This should be thought of in the HTTP 1.1 Request/Response sense although in some cases this will not be the wired/wireless protocol used to deliver the message.

#### When to use

The DMI message processor calls the methods of this interface.

#### Methods in V-table Order

Interface::Methods	Description
<b>IUnknown::QueryInterface</b>	Returns pointers to supported interfaces.
<b>IUnknown::AddRef</b>	Increments reference count.
<b>IUnknown::Release</b>	Decrements reference count.
<b>IMessageHandler::OnMessage</b>	Called when the message processor has a message to process.
<b>IMessageHandler::AcceptFormats</b>	Returns an array of the formats accepted. This method is called by the DMI when it needs to determine proper message handling.
<b>IMessageHandler::ProduceFromats</b>	Returns an array of the formats generated. This method is called by the DMI when it needs to determine proper message handling.
<b>IOutboundConnector::GetCarrierInfo</b>	Called by the DMI to retrieve the carrier information in an <b>IInfoData</b> object.

## 2.8 LOCALIZATION

All strings that are not “debug only” shall be made localizable. The main components that must be localizable are the Event logging and Performance counters but, as the coding proceeds the developers should be aware that this is a issue and should ask the question “Does this string need to be localized?” every time that a string is coded.

## 2.9 REGISTRY ENTRIES

The following table outlines all the registry entries that are used by the DMI. This is intended to be a comprehensive list and includes a column identifying those entries that are to be made public and those entries that are private. Private entries are defined to be those entries used by the system developers and test engineers to aid in development or test. Private entries are defined to be those entries guaranteed not to change from version to version or are chosen to be exposed to the public.

#### Key to the table:

**PATH** – Gives the full path to the entry

**K/V** – Represents whether the entry is a key or a value. Where K == Key and V == value.

**Type** – Represents the data type associated with the entry. Where DW == REG\_DWORD, SZ == REG\_SZ, MSZ == REG\_MULTI\_SZ, ESZ == REG\_EXPAND\_SZ, B == REG\_BINARY

**Use** – Represents the usage code. Where P == Public to be exposed external to Microsoft, D used by development, T used by test.

**Description** – Represents a brief description of the entry purpose.

Path	K / V	Type	Use	Description
HKLM\Software\Exchange\DMILogging	V	DW	P	Defines the logging level to use. Currently only level 0 and 1 are defined. If this value is not present the default is level 0. If this value is other than 0 the level is 1.

Path	K / V	Type	Use	Description
HKLM\Software\Exchange\DMI\OptimumMessageSize	V	DW	P	Defines the optimum message size. This governs the creation of IMessageObjects and the amount of memory that is initially allocated to them. It should be set to some default size at installation time.

## 2.10 DEBUG FRAMEWORK

The debug framework consists of a set of header files that contain macros for debugging. These macros should completely remove themselves from the retail builds of the code.

The macros should allow for a standard format that is extensible when needed and yet easy to use. Since the DMI is a NT service the debug data should sent to the system debug window and optionally to the event log or some other log file.

The debug code should be placed into the exception handling code such that all exceptions that are thrown can be tracked to their source.

An optional feature to all debug macros should be the ability to return the file name and line number. This should be controllable from a global perspective as well as a local perspective. In other words I should be able to say, “for this debug statement always return file/line information” and have a global setting that says “return all file/line information regardless of local settings”.

## 3. Design Options

There are many design options that were reviewed. This section is a collection of the options that covered and the rationale that went into the choice that was made. <Much of this section has been intentionally removed at the time of this snapshot>

### 3.1 THREAD POOLING

The DMI needs thread pool for use by the server. The threads in the pool will be used to handle message traffic and thus the handling of these threads is central to the operation of the server. In addition each thread may need to take on the SID of a different user account or group allowing the thread to impersonate a user other than the system. The basic requirements of this function are;

**Reliability** – The code that handles the thread pool and worker thread activation and deactivation must be robust and allow for the case where the device module loaded hangs.

**Multi-processor friendly** – The thread code must work in a multi processor system as well as a single processor system.

**Queuing of waiting work** – The queuing should be efficient enough such that the system does not spend a significant amount of time in starting a waiting task.

**Thread statistics** – Maximum concurrent threads, Average run time, Maximum run time

### 3.2 INBOUND TCP CONNECTIONS

The system DMI will need a code base that handles the inbound TCP request traffic. This functionality will essentially emulate the client connection to the HTTP source server (the may be Exchange through DAV-EX or IM or a WEB server.)

The following is a set of requirements:



**Reliability** – The code that handles the TCP connections and HTTP request and responses must be reliable. There must be no handle or memory leaks. It must be able to handle a wide range of HTTP responses and errors with out hanging.

**Scaleable** – This code must be able to handle many concurrent connections and this must be a settable feature.

### 3.2.1 IIS

I believe that this is a given and that little in the way of pros and cons need to be covered. It does not make sense to not use the IIS by way of ISAPI. With that said the only question here is do we code our own ISAPI DLL from scratch or to we begin by leveraging existing code from the Exchange team. The previous section for a discussion of the merits of using code from other teams.

## 3.3 OUTBOUND UDP CONNECTIONS

The main issue here is “Do we support UDP connections to the DMI?”

Since Exchange is the only server that we know of at this time that will be sending HTTPU data, and that is only for subscriptions, we need to determine if this is a feature that we wish to support.

**Reliability** – The code that handles the UCP connections and initial filtering of HTTPU syntax parsing must be reliable. There must be no handle or memory leaks. It must be able to handle a wide range of HTTP syntax errors with out hanging and return the appropriate response to the requester.

**Scaleable** – This code must be able to handle many concurrent connections and this must be a settable feature.

## 3.4 INTERNAL QUEUES

The system will need some internal queues. Some of the queue requirements are:

**Reliability** – The queues must never drop messages, fail to receive a message with out notification to the caller. The queues must never have incomplete item in the queue (i.e. transactions must be atomic)

**Scalability** – The queues must allow the system to scale to many messages per second in a system. This may mean that the queues may have to be shared across process and/or machine boundaries.

**Persistence** – The queues must allow for both persistence and non-persistent modes of operation. In other words if the system goes down anything in the queue is maintained or lost depending on the programmatic state of the queue.

**Entry Time-out** – The queues must allow for items in the queue to time out with a call back response (or some method of checking if the item has timed-out). (This is for the case where we need to handle idle time on TCP connections.)

**Queue statistics** – Input rate, Input count, Output rate, Output Count, Max queue size, Average queue Size, Max Time in queue , Average Time in queue and more as the need may arise. It should be noted that these are queue centric statistics.

## 4. References

[1] J Mogul, Tim Berners-Lee, Larry Masinter, P. Leach, R. Fielding, H. Nielsen, Jim Gettys, "Hypertext Transfer Protocol -- HTTP/1.1", 09/11/1998. <http://www.ietf.org/internet-drafts/draft-ietf-http-v11-spec-rev-05.txt>

[2] D. Kadyk, H. Yu, “DMI and third party software handling”, 11/24/1998  
<http://exchange/doc/specs/Platinum/Voice%20and%20Wireless%20Integration/ThirdPartySWHandling.doc>

[3] H. Yu, “DMI Wireless Security”, 11/21/1998  
<http://exchange/doc/specs/Platinum/Voice%20and%20Wireless%20Integration/DMI%20Wireless%20Security.doc>

## 5. Open Issues

- *<this section removed for the purpose of this snapshot>*

## **EXHIBIT F**



General

Summary

Statistics

Contents

Custom

Created: Monday, May 03, 1999 8:34:00 AM

Modified: Friday, June 02, 2006 11:38:14 AM

Accessed: Tuesday, June 13, 2006 5:00:26 PM

Printed: Tuesday, May 04, 1999 3:38:00 AM

Last saved by: eschultz

Revision number: 46

Total editing time: 1179207497 Minutes

Statistics:

Statistic name	Value
Pages:	11
Paragraphs:	159
Lines:	988
Words:	3226
Characters:	18766
Characters (with spaces):	21981

OK

Cancel

# **Exchange Wireless Overview**

**Wireless Team**  
**Exchange Enhanced Platform Product Group**  
**Monday, April 26, 1999**

**MICROSOFT CONFIDENTIAL**

Contact: marcse or eschultz

<b>INTRODUCTION .....</b>	<b>3</b>
<b>VISION – MICROSOFT’S END-TO-END WIRELESS SOLUTION.....</b>	<b>3</b>
MICROSOFT ASSETS .....	3
AREAS OF FOCUS.....	4
TRENDS .....	5
<b>PLATINUM WIRELESS SERVICES.....</b>	<b>5</b>
FUNCTIONALITY .....	5
<i>End-User</i> .....	5
<i>Administration</i> .....	6
<i>Extensibility</i> .....	6
DESIGN .....	6
<i>Information Gatherer</i> .....	7
<i>Information Gateway</i> .....	7
<b>CUSTOMERS .....</b>	<b>8</b>
<b>COMPETITION .....</b>	<b>8</b>
<b>STRATEGY .....</b>	<b>9</b>
ROADMAP .....	10
PROGRESS .....	10
MISSING ELEMENTS.....	11

## Introduction

Wireless connectivity and the mobile user represent a significant, fast developing market which promises to soon eclipse in number wired connections and conventional desktop computers. This trend presents both an opportunity for Microsoft to align its assets to provide a compelling solution to customers, as well as a potential discontinuity and a profound threat to our business.

We are witnessing the convergence and climax of several enablers resulting in an explosive market: *airlinks* provide coverage with reasonable bandwidth data services; *devices* are available at reasonable price points with acceptable form factor, power consumption, processor, memory, and storage; *carriers* are providing compelling pricing while competition and growth drivers are forcing them to look to differentiation and new services; *end-users* are more sophisticated as a result of their wired experience and are both more aware and more dependent on their email and other information.

In the U.S., we are at some disadvantage as these developments are more advanced in other countries and regions where cell phone penetration is higher and users and services more sophisticated, e.g. Scandinavia and Japan. Early wireless data services such as email are already widely available in these regions.

Competition is forming rapidly, mostly as alliances, which threaten every element of our value proposition. The most notable competitors include IBM/Nokia Wireless Notes, Yahoo! Online Everywhere, Palm, AOL, Phone.Com, Symbian and WAP. There is a risk that we will lose thought leadership and real markets. The objective of this paper is to provide an overview of our understanding of this space, as well our plan to succeed.

## Vision – Microsoft’s End-to-End Wireless Solution

Wireless allows users to be untethered, but still connected. Mobile users are able to access time critical information and communicate anytime, anywhere increasing productivity and the velocity of information and decisions.

*To enable users of mobile devices to access their personal information (e.g. email, calendar, contacts), as well as private and public information sources and transaction services over a variety of airlinks using Microsoft products and technologies.*

Platinum Wireless Services provides both one and two-way communication between device and corporate intranet allowing information to be “pushed” to the device as notifications or updates, or requested by the device as in the case of browsing.

Microsoft has many of the assets required to build and deliver an end-to-end solution with Platinum as the platform and the delivery backbone. We must align these assets, invest in building and buying those components required to complete the solution, integrate and execute.

## Microsoft Assets

Our assets include:

- Exchange / Platinum – the core and backbone of the wireless platform and the center of users’ personal information. Personal information – email, calendar, tasks, contacts – is cited as the most critical asset and one for which users would be willing to pay. We are building on the Exchange asset to provide personalization, as well as, notification and synchronization services and a communication backbone extensible to a wide array of information sources,

end-point devices and airlinks. We are also determining how best to integrate our Wireless and Unified Messaging efforts to maximize synergy.

- WinCE – client platform providing initial entry with OS-independent micro-browser and scaling to full WinCE OS and application suite.
- MSN, HotMail – content, information and transaction services, as well as consumer mail platform.
- Tools – allows rapid innovation of our wireless platform by creating new applications and services.
- Customers – our installed corporate users provide a well-targeted market for introducing wireless access of essential information.
- Partners – devices manufacturers to deploy our client solutions, carriers to deploy our server platform and support our client device solution, ISVs and systems integrators.

## ***Areas of Focus***

In Platinum Wireless we are focusing on the following areas:

- **Registration** - New devices will be added to the directory and we will record essential information on the device and connection. This will allow us to optimize services and content for the user's device. We will also incorporate presence information to optimize delivery.
- **Personalization** - Users will be able to subscribe to specific information sources and provide reasonably fine filters and rich views for information accessible from their devices. The user will administer personalization information using a desktop device and a full web interface. We will leverage the events and filtering provided by Platinum to implement the actual information gatherer expressing a user's interests as a set of subscriptions against a set of Platinum folders. Our "Information Gateway" provides an interface (HTTP extensions) to allow access of other information sources, as well as participation by proactive and interactive agents.
- **Access** – Information will be delivered to devices of widely ranging capabilities. We must accommodate low-end devices such as simple one-way pagers, which communicate through very small packets containing a small alphanumeric payload, up through high-end devices such as tablet computers running Outlook. We are building a delivery system for information (Information Gateway) which accommodates these devices through device-class specific modules. These modules can render and reformat information, add synchronization tokens and implement specific optimizations. These modules are implemented as COM objects. We will deliver a required set of these modules and provide interfaces to allow partner and vendors to create additional modules.
- **Network / Transport** – Information will be delivered over various transports and airlinks. These transports represent different standards and protocols, addressing, security and performance. We will provide a mechanism to bind and communicate through specific transports using network-class specific modules implemented as COM objects. We will deliver a required set and provide interfaces for partners and vendors to create additional modules.



## **Trends**

Our solution must work with the current deployed infrastructures, the immediate-term devices and must embrace the likely developments of the next five years:

- *carriers*: higher penetration rates, better coverage, lower airtime costs.
- *airlinks*: rapid conversion from analog to digital, faster, data-capable moving from today's 9.6/14.4 kbps to 2.5G (GPRS, HDR) with 64kbps and beyond.
- *micro-cell*: in-building wireless LAN, micro-cell in addition to wide-area cell.
- *roaming/switching*: universal roaming between carriers and countries, and seamless switching between micro and wide-area.
- *identity*: security and personalization – which follows the user (e.g. SIM card).
- *devices*: high replacement rate, rapid innovation, more resources culminating in the smart-phone and the tablet computer.
- *standards*: end-to-end internet standards. This will allow us to extend the connected universe while leveraging all of our standards-based assets.

These developments will provide continuity between “wired” and “un-wired”, bringing them together as simply “connected”. It is our responsibility to create a compelling end-to-end solution to ensure this happens on Microsoft products and technologies.

## **Platinum Wireless Services**

### **Functionality**

#### **End-User**

End-user functionality includes various means of accessing data and configuring the device preferences.

- **Accessing Data** – Platinum Wireless Services provide both one and two-way communication between device and corporate intranet allowing information to be “pushed” to the device as notifications or updates, or requested by the device as in the case of browsing. The following modes for accessing data are supported by devices and airlinks, as appropriate.
  - **Browse** – This is a slight misnomer as it implies that users on small form factor mobile devices can “browse” random web content in a meaningful way. Browsing simply means that the primary interface is a web (micro)browser, and that the information is formatted as HTML with links and references. Any meaningful web content will likely have been rendered specifically for restricted device footprints.
  - **Notification** – Information is “pushed” to the device by the Information Gatherer or an Agent. The information is formatted specifically for the device and may include the message header, some form of message contents and optionally a reference or link used to retrieve the rest of the message.
  - **Update / Synch** – Certain mobile devices have applications which store elements such as emails, contacts and calendar items and identify these items unique to support synchronization. In these cases, the specific device module can include synch information (token or identifier) with the “pushed” information..

- Request – The device can issue a request, which is filled by some agent or application listening for the request.
- Personalization – designating the information in which the user is interested, configuring the way in which the information is shown on specific devices (views, filters) and setting various other preferences. Users will likely configure personalization options for devices using a full desktop client and a web page interface and unlikely that they will use a restricted device such as a cell phone with a limited keyboard for this task.

## **Administration**

Administrative functions include the following:

- Registration – adding a new device and its capabilities to the directory and binding the device instance to a particular user.
- Provisioning – adding additional services to a device, querying account specifics such as minutes used on plan.

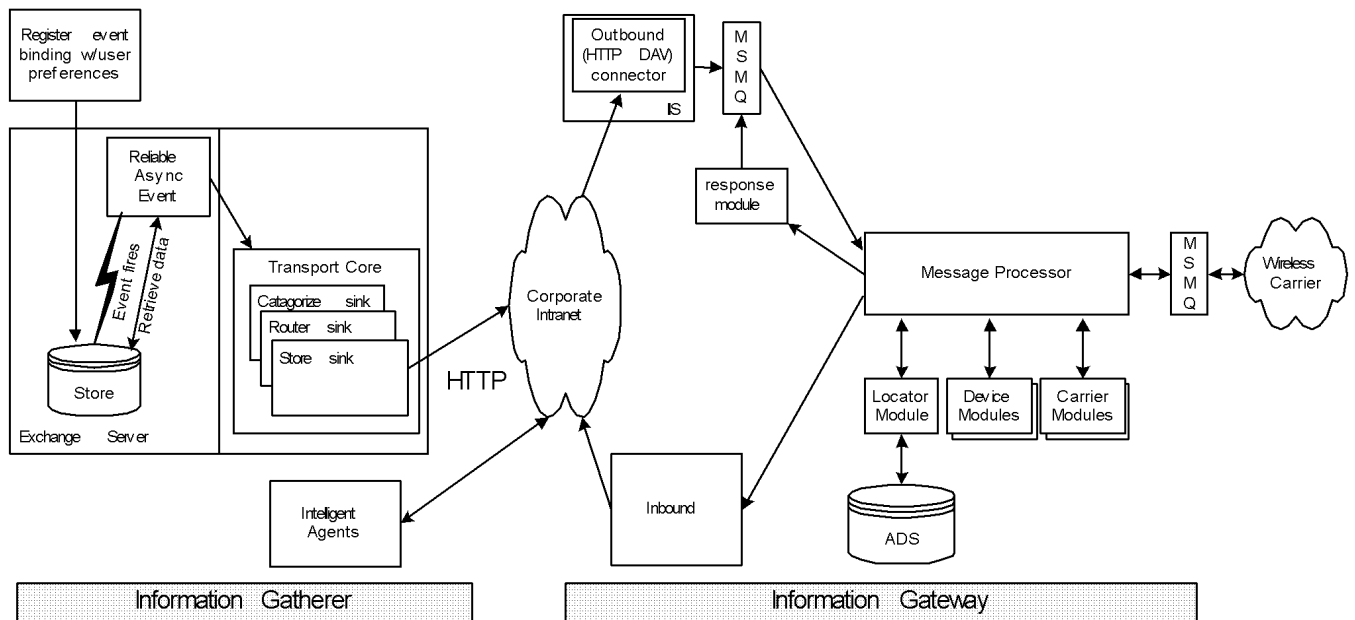
## **Extensibility**

Platinum Wireless Services provides mechanisms for adding and extending functionality.

- Intelligent Agents – Applications can be created which push information to devices, or respond to requests initiated by devices using the HTTP interface to the Information Gateway. These agents can be either proactive, initiating a “push” to the device based on triggers or schedules, or interactive, responding to requests from the device.
- The modules below are all COM objects loaded and managed by the Message Processor with a defined interface.
  - Device Module – add support for new device type and capabilities
  - Carrier Module – add support for a new type of carrier
  - Optimization Module – add any specific message processing such as Encryption, Authentication, Compression, etc.

## **Design**

Platinum Wireless services are implemented as two primary components. The Information Gatherer reacts to changes in the Platinum store and forwards HTTP notifications to the Information Gateway. The Information Gateway provides bi-directional communication between specific devices and the corporate intranet over specific carrier connections.



## Information Gatherer

- Events – reliable async events are the basis of registering user preferences and providing efficient, filtered notification services for Platinum information.
  - Registration – events are registered against specific folders with “where” clauses
  - Firing – events are consumed by the Transport Core
- Transport Core – processes the events and emits HTTP notifications to the Information Gateway
- Intelligent Agents – Agent processes can be added which communicate with the Information Gateway using the standards-based HTTP interface. These agents can be interactive, responding to user requests (e.g. get information on specific stock symbol, or conducting a transaction) or proactive, monitoring and collecting information and sending notifications and updates to users (e.g. sending scheduled compilation of periodic portfolio performance). Agents of either type could be used to deliver content such as MSN.

## Information Gateway

- Outbound Connector (HTTP) – ISAPI extension which sinks the HTTP notification and forwards on to Message Processor through MSMQ
- Inbound Connector – COM module which bridges the process boundary from the Message Processor and forwards on through HTTP.
- Response module – COM module which bridges the process boundary from the Message Processor and hands response back to OutBound connector via MSMQ.
- Message Processor – Provides two-way processing of messages, including loading and maintaining the correct chain of Device and Carrier modules.
  - Locator module – provides address translation and setup of Message Processor modules based on user and device registration.

MICROSOFT CONFIDENTIAL

Contact: marcse or eschultz

- Device module – performs processing and formatting of information for specific devices.
- Carrier module – provide the basic interfacing between the Message Processor and a specific carrier network.
- Encryption, Authentication, Optimization module (not shown) – The Message Processor can also load special purpose modules to implement such functions as encryption, compression and authentication.
- ADS – Active Directory. Stores user and device registration information which is used for address translation and selecting appropriate device and carrier modules.

## Customers

The primary customers for our wireless services are corporate and consumer end-users, as well as carriers who market the service and host the data center, the device manufacturer and various software vendors.

- Corporate – includes both enterprise customers (LORG, SME) which require access to their personal mail and corporate information, as well as small business users which required hosted access and service. These users are most willing to pay for access to their critical personal and business information.
- Consumer – consumers represent the largest segment of mobile users in many markets, though difficult to monetize. We will provide HotMail and MSN services for consumer users.
- Carrier – The carrier is highly motivated to offer differentiated, value-added services to assist in the costly business of subscriber acquisition and subscriber retention, as well as generating incremental revenues for services and airtime. The carrier also stands to reduce customer service costs through subscriber self provisioning and profile maintenance.
- ISVs, IHVs – The device manufacturers are motivated to support advanced services in order to differentiate their advanced devices from commodity devices, and to build a volume market. These devices are generally higher margin but require volumes in order to defray the costs of design and production.

## Competition



- [Redacted] with free
- ch
- 
- all
- 

## Strategy

- Th  
cu  
ef  
w/  
**Pi**  
ex  
ba  
an  
to
- [Redacted] late n
  - pur
  - it-
  - of
  - cess
  - n can
  - tical
  - ver
  - rs for
  - in
  - phurn
  - d data
  - omers
  - n and
  - t wait
  - to
- CY00.

- D [Redacted] ed
- P an ap w in late tify

## Roadmap

We expect the following developments over the next three years:

1999	2000	2001
<ul style="list-style-type: none"> <li>• Commercial deployments</li> <li>• Current generation products</li> <li>• Basic handsets</li> <li>• Basic services – email, forward to fax, data query</li> </ul>	<ul style="list-style-type: none"> <li>• Increased deployments</li> <li>• Platinum-based offering</li> <li>• Next generation handsets with new MicroBrowser</li> <li>• Secure access to corporate data</li> <li>• Access to general web content (MSN Mobile)</li> <li>• Synch to Palm Sized PC (CE)</li> </ul>	<ul style="list-style-type: none"> <li>• First 2.5G packet data networks (GPRS)</li> <li>• Richer services</li> <li>• Unified messaging</li> <li>• Vertical applications</li> <li>• Increased value proposition and pricing</li> </ul>

## Progress

- We dev [Redacted] ll as
- -
- In
- rent
- can others
- ic

[Redacted]

ial  
crest. These  
com and

### ***Missing Elements***

In o  
thro

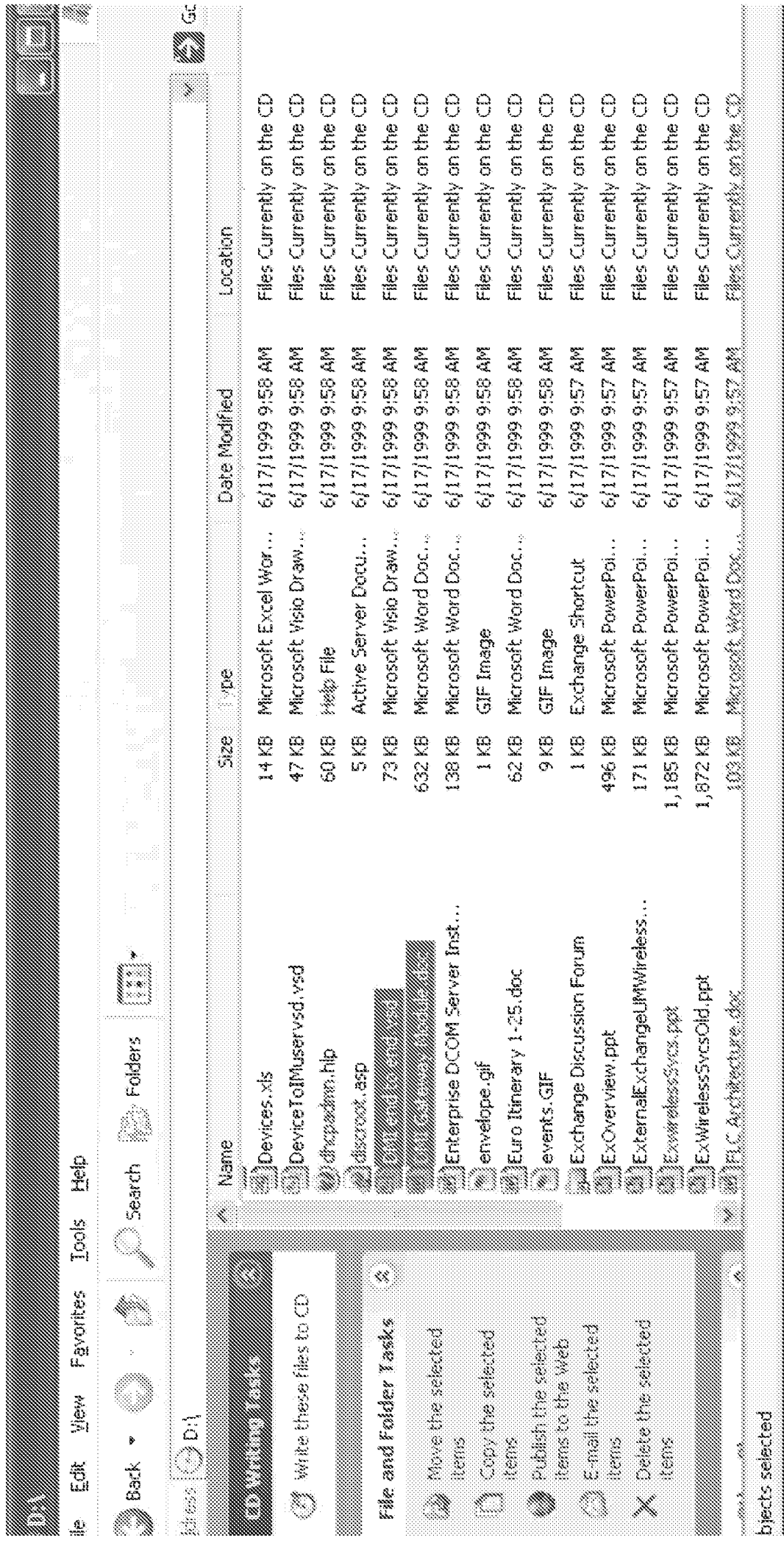
[Redacted]

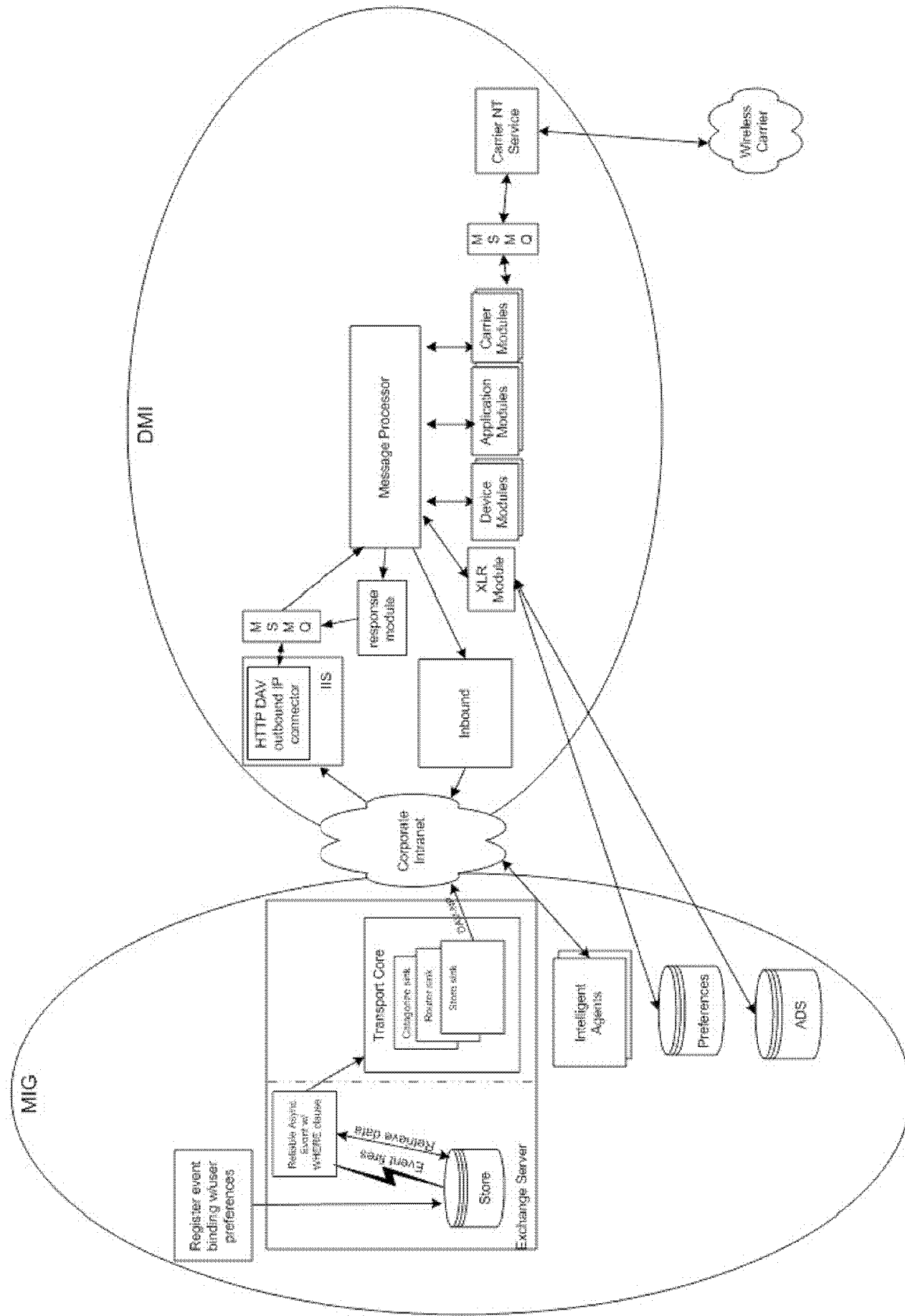
either

- 
- 
- 
- 
-

## **EXHIBIT G**







## **EXHIBIT H**

Airstream Carrier Alpha Release  
4524.0



FOR INTERNAL MICROSOFT USE ONLY

Contents are property of the Microsoft Corporation.

**Microsoft**

WINDOWS BURN LAB  
REDMOND  
Confidential

# SETUP.EXE Properties



General

Compatibility



SETUP.EXE

Type of file: Application

Description: SETUP

Location: D:\

Size: 16.2 KB (16,656 bytes)

Size on disk: 18.0 KB (18,432 bytes)

Created: Thursday, August 03, 2000, 1:00:00 AM

Modified: Thursday, August 03, 2000, 1:00:00 AM

Accessed:

Attributes:



Read-only



Hidden

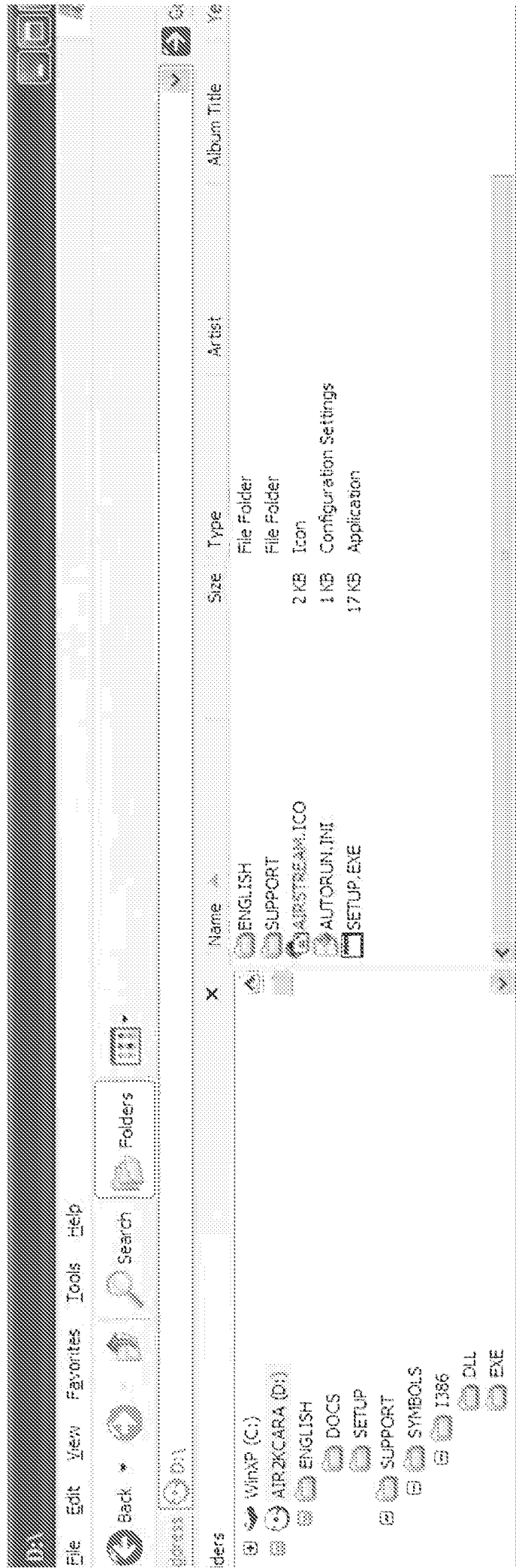


Archive

OK

Cancel

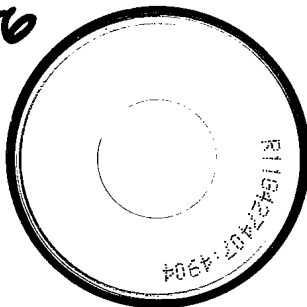
Apply



## **EXHIBIT I**

MIS2001CAR

9/21/00  
4556



FOR INTERNAL MICROSOFT USE ONLY.

Content is the property of the Microsoft Corporation.

**Microsoft**

WINDOWS BURN LAB  
REDMOND  
Confidential



MS2001CAR (D:)				
DOCS				
SETUP				
PROGRAM FILES				
AIRSTREAM				
AIRWEB				
IMAGES				
USA				
COMMON FILES				
MICROSOFT SHARED				
CDO				
SYSTEM32				
REDIST				
MS				
SYSTEM				
TEMP				
SPANISH				
DOCS				
SETUP				
PROGRAM FILES				
AIRSTREAM				
AIRWEB				
IMAGES				
SPA				
COMMON FILES				
MICROSOFT SHARED				
CDO				
SYSTEM32				
REDIST				
MS				
SYSTEM				
TEMP				
SUPPORT				
SYMBOLS				
1386				
DLL				
EYE				

DOCS				
SETUP				
AUTO_EN.DBD				
CLOSING.BMP				
DENO32.EXE				
D32.DLL				
LAUNCH.EXE				
LAUNCH.INI				

File Folder				
File Folder				
DBD File	20 KB			
Bmp File	15 KB			
Application	364 KB			
Application Extension	28 KB			
Application	9 KB			
Configuration Settings	1 KB			

## **EXHIBIT J**

Microsoft Mobile information 200:  
Server Enterprise Edition  
Beta 1 Release  
10/09/2000  
Build 4556.7



FOR INTERNAL MICROSOFT USE ONLY  
Contents are property of the Microsoft Corporation.

**Microsoft**

WINDOWS BURN LAB  
REDMOND  
Confidential

